Tiago Alexandre Gonçalves Alvané

Navegação de Drones via GPS

(*Drones Navigation through GPS*)

Tiago Alexandre Gonçalves Alvané

Navegação de Drone via GPS

(Drone navigation through GPS)

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Nuno Miguel Gonçalves Borges, Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e coorientação do Professor Doutor João Nuno Pimentel da Silva Matos, Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho aos meus pais, António e Margarida, por todo o esforço e dedicação.

"Our virtues and our failings are inseparable, like force and matter. When they separate, man is no more."

Nikola Tesla

o júri / the jury

presidente / president

Prof. Doutor José Carlos Esteves Duarte Pedro
Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Nuno Miguel Gonçalves Borges de Carvalho (Orientador)
Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Doutor Rafael Ferreira da Silva Caldeirinha
Professor Coordenador do Instituto Politécnico de Leiria

Agradecimentos/ Acknowledgements

Dedico este trabalho a todas as pessoas que duma forma ou de outra me acompanharam em todo este percurso e que fizeram com que este fosse concluído com sucesso.
Em primeiro lugar quero agradecer aos meus pais e irmã por terem sido incansáveis no apoio, no amor e na coragem que me deram ao longo deste meu percurso. À minha namorada, Ana Luísa, um enorme obrigado por ter estado ao meu lado, transmitindo sempre a confiança e o amor necessário para que todas as etapas e obstáculos fossem ultrapassadas com sucesso.
Aos meus avós, Emília e Alexandre, um muito obrigado pelo apoio e por serem uma fonte inspiração onde o esforço e a determinação são denominadores comuns.
A todos os meus amigos e colegas com os quais tive um enorme gosto de trabalhar e cooperar ao longo do meu percurso académico, obrigado por fazerem parte deste trilho fantástico que percorremos e com o qual sorrimos, aprendemos, discutimos, brincamos e crescemos juntos.
Gostaria de agradecer em especial ao Professor Doutor Nuno Borges de Carvalho, por me ter cativado e me ter orientado pelo caminho certo, mostrando-se disponível sempre que assim o necessitei.
Queria também agradecer a todos os investigadores e colabores do Instituto de Telecomunicações, em especial à equipa de RF com os quais tive um enorme gosto de privar e passar bons momentos.

A todos vós, um profundo muito obrigado!

Palavras-chave

Drone, *Quadcopter*, Ar. Drone 2.0, GPS, Navegação autónoma.

Resumo

Atualmente vivemos numa era onde a evolução tecnológica cresce exponencialmente e onde novos conceitos surgem diariamente para gáudio de uns e desagrado de outros. Uma das tecnologias mais emergentes e com maior impacto nos últimos anos é a comercialização de *quadcopters* de pequena dimensão, também denominados de *drones* pertencendo à categoria de *Unmanned Aerial Vehicle* (UAV). A possibilidade de estes poderem ser tripulados através de controladores capacitados de comunicações rádio ou mais recentemente por simples *tablets* ou *smartphones*, fez com que o interesse de aquisição atingisse valores sem precedentes, ao ponto de os departamentos de defesa e segurança de vários países sentirem a necessidade de legislar, de modo a preservar a segurança e privacidade. Esta necessidade prede-se com o facto de a maioria dos *drones* comerciais permitem alcançar uma altitude considerável e estão maioritariamente equipados com câmaras com alta definição. Com a comercialização de *drones* das mais variadas dimensões, não tardaram a surgir uma infinidade de aplicações civis para cenários de qualquer índole. À medida que as aplicações surgiram, surgiu também a necessidade de integração de vários sensores que contribuíssem para um voo mais controlado, libertando o piloto da necessidade de um controlo total sobre o *drone.* A inclusão de tecnologias de localização GPS surgem assim como uma mais-valia no controlo através o posicionamento geográfico, tal como a possibilidade de serem realizados voos completamente autónomos, ficando do lado do utilizador a decisão de definir o trajeto a realizar pelo *drone*, sendo possível a introdução de vários pontos de passagem (*waypoints*) ao longo de um percurso definido. Como tal, o objetivo do trabalho desenvolvido e descrito neste documento foi o de desenvolver uma aplicação que permitisse a introdução de *waypoints* em formato de coordenadas geográficas em graus decimais, ficando do lado da aplicação a capacidade de receção de mensagens de localização através de um recetor GPS, análise das variáveis de controlo do *drone* e envio de comandos de navegação. São ainda gerados algoritmos de segurança, salvaguardando assim o sistema de possíveis erros de leitura ou falhas de comunicação.

keywords Drone, Quadcopter, Ar. Drone 2.0, GPS, Autonomous flight.

abstract

Nowadays, we live in an era where the technological evolution is growing exponentially. New concepts come out daily for the delight of some and the displeasure of other. In the last two years, one of the most emerging technologies with a huge impact on people's life was the put up for sale of quadcopters which is a class of Unmanned Aerial Vehicles (UAV) and also known as drones. The fact that they can be controlled by a RC command or more recently by tablets or smartphones resulted in an unprecedented rise of the drone's purchase. In many countries, this reality forced the defense and security departments to be alert and create laws to ensure the security and privacy of everyone. This need to protect the privacy and security stemmed from the fact that most recent drones are provided with high definition cameras and can achieve a considerable altitude. Due to the drone's commercialization with the most varied dimensions, many civil applications were taking shape for any type of scenarios. As new applications emerged, also came up the need for the integration of sensors to allow smoother flights with a better control, liberating the pilot from the need to take full control of the drone. Thus, the inclusion of navigation systems, such as GPS, becomes a very powerful tool. This technology allows a better control over the location of the drone, as well as the option of fully autonomous flight. This feature made the piloting task easier, being only essential to define the flight path through the introduction of waypoints. Therefore, the purpose of the work developed and presented in this document is to develop an application which allows the instruction of waypoints coordinates in decimal format. The application receives coordinates from a GPS receiver, analyses the control variables and sends the navigation commands to the drone. The application also contains a security system, to circumvent eventual communication failures.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

AGPS – *Assisted Global Positioning System*

AMCS – *Alternate Master Control Station*

API – *Application Programming Interface*

ARM – *Architecture RISC Machine*

ASCII – *American Standard Code for Information Interchange*

AVL – *Auto Vehicle Location*

BDS – *BeiDou Satellite Navigation System*

BPS – *Bits per Second*

C/A – *Coarse Acquisition*

CES – *Consumer Electronic Show*

CMOS – *Complementary Metal-Oxide Semiconductor*

COM – *Component Object Model*

CR – *Carriage Return*

CRC – *Cyclic Redundancy Check*

DDR – *Double Data Rate*

DGPS – *Differential Global Positioning System*

DHCP – *Dynamic Host Configuration Protocol*

DOD – *Department of Defense*

DOF – *Degrees Of Freedom*

EEPROM – *Electric-Erasable Programmable Read-Only Memory*

EGNOS – *European Geostationary Navigation Overlay Service*

EPO – *European Patent Office*

ESA – *European Space Agency*

FAI - *Fédération Aéronautique Internationale*

FCC – *Federal Communications Commission*

FIFO – *First In First Out*

FPS – *Frames per Second*

FTDI – *Future Technology Devices International*

FTP – *File Transfer Protocol*

GA – *Ground Antennas*

GLONASS – *Global Navigation Satellite System*

GNSS – *Global Navigation Satellite System*

GPS – *Global Position System*

HDOP – *Horizontal Dilution of Precision*

IC – *Integrated Circuits*

ID – *Identifier*

IDE – *Integrated Development Environment*

IEEE – *Institute of Electrical and Electronics Engineers*

IMU – *Inertial Measurement Units*

IP – *Internet Protocol*

LBS – *Location-based service*

LED – *Light-emitting Diode*

LF – *Line Feed*

MAV – *Micro Aerial Vehicle*

MCS – *Master Control Station*

MEMS - *Micro-Electro-Mechanical System*

MISO – *Master In Slave Out*

MOSI – *Master Out Slave In*

MS – *Monitor Station*

NAVSTAR – *Navigation Satellite with Time and Ranging*

NGS – *National Geodetic Survey*

NMEA – *Nation Marine Electronics Association*

NNSS – *Navy Navigation Satellite System*

OSI – *Open Systems Interconnection*

P-code – *Precision Code*

POT – *Patch on top*

PRN – *Pseudorandom Noise*

PWM – *Pulse Width Modulation*

QCIF – *Quarter Common Intermediate Format*

RAM – *Random Access Memory*

RC – *Radio Controlled*

RINEX – *Receiver Independent Exchange Format*

RISC – *Reduced Instruction Set Computing*

RMS – *Root Mean Square*

RoHS – *Restriction of Certain Hazardous Substances*

RPM – *Rotations per Minute*

RTC – *Real Time Clock*

RTCM SC – *Radio Technical Commission for Maritime Services Special Committee*

SBAS – *Satellite-based Augmentation System*

SCK – *Serial Clock*

SCL – *Serial Clock*

SDA – *Serial Data Line*

SDK – *Software Development Kit*

SO – Sistema Operativo

SRAM – *Static Random Access Memory*

SS – *Slave Select*

TCP – *Transmission Control Protocol*

TTL – *Transistor-Transistor Logic*

VANT – Veiculo Aéreo Não Tripulado

VGA – *Video Graphics Array*

UART – *Universal Asynchronous Receiver/Transmitter*

UAV – *Unmanned Aerial Vehicle*

UDP – *User Datagram Protocol*

USB – *Universal Serial Bus*

WP - *Waypoint*

Capítulo 1

Introdução

1.1 Motivação

No panorama tecnológico atual, os *quadcopters* ou também conhecidos como *drones* estão a atingir um patamar de tal forma relevante, que deixaram de ter um reconhecimento equiparável a um objeto perigoso utilizado para fins militares, passando a serem vistos com um objeto comum para seres comuns. Este facto veio contribuir para a idealização de um número infindável de aplicações civis, devido a serem vistos como objetos úteis em inúmeras manobras que poderiam colocar em risco de vida de um ser humano, ou simplesmente devido a proporcionarem benefícios na realização de determinadas tarefas tanto a nível de dificuldade operacional como a nível monetário.

Existem diversos exemplos em que os *drones* foram e são utilizados com sucesso. Casos como a avaliação dos estragos nos reatores nucleares danificados na central de Fukushima [1] no desastre de Março de 2011 ou a prevenção e investigação de incêndios [2], são apenas dois exemplos de sucesso das potencialidades a reter.

Surgem então, vários cenários de aplicações civis [3] onde estes poderão desempenhar um papel importante, entre eles:

- Aplicações de proteção civil
    - Vigilância contra atos ilegais na costa marítima;
    - Vigilância de florestas contra incêndios;
    - Criação de redes de comunicações de emergência;
    - Controlo de fronteiras;
    - Controlo e rastreamento contra cultivo de drogas ilegais;
    - Vigilância de eventos desportivos ou sociais com um elevado risco de confrontos físicos.

- Investigação:
    - Investigação arqueológica;
    - Medição de possíveis contaminações em ambientes de difícil acesso humano;
    - Investigação de glaciares ou oceanos;
    - Identificação de novas espécies de animais ou plantas.
- Captação de vídeo e fotografia, permitindo realizar um registo de imagem para vários tipos de desígnios:
    - Contagem de aglomerados de animais;
    - Monotorização de florestas;
    - Eventos sociais públicos ou particulares.
- Inspeções industriais:
    - Inspeções de oleodutos ou gasodutos;
    - Inspeções de linhas de energia elétrica ou postes de alta tensão;
    - Manutenção de eólicas ou pontes.

Em suma, existe uma panóplia de tarefas de difícil execução, onde os *drones* se revelam ser uma ferramenta indispensável.

Historicamente, o primeiro *quadcopter* surgiu em 1920 num projeto da Força Aérea Americana [4]. Após várias tentativas, Etienne Oehmichen apresentou o seu segundo, composto por quatro motores de duas pás, controlados por um único motor, presente na Figura 1.1. A aeronave apresentava uma estabilidade e controlabilidade considerável para a época.

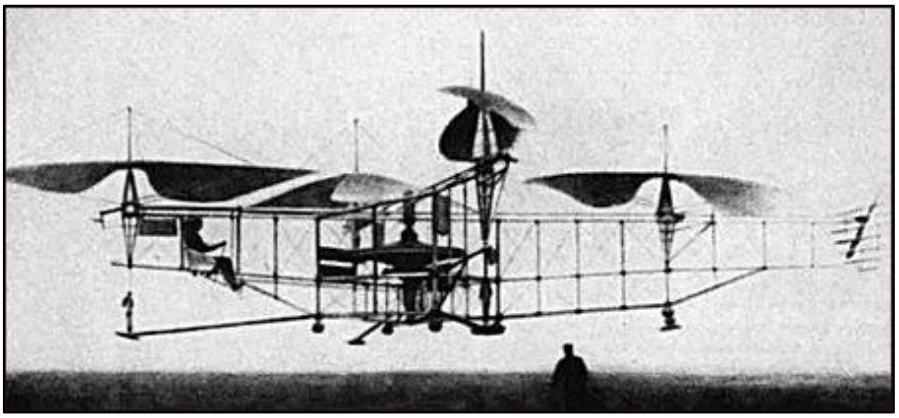

Figura 1.1 Segundo *quadcopter* de Etienne Oehmichen [7]

Realizou mais de mil voos e registou o primeiro recorde de distância da *Fédération Aéronautique Internationale* (FAI) em 360 metros [5] para um helicóptero, batendo esse próprio recorde três dias mais tarde em 525 metros [6].

Com o surgimento dos helicópteros convencionais de dois motores, com uma menor complexidade mecânica e maior estabilidade e controlo relativamente aos *quadcopters*, fez com que estes caíssem em desuso deixando de ser um foco de interesse durante décadas.

Na última década, devido à investigação desenvolvida em redor dos *quadcopters*, estes recuperaram a sua popularidade, tornando-se numa aeronave muito mais simples de controlar do que os helicópteros convencionais.

Existem diversos tipos de UAVs, sendo que um dos mais comuns e dos mais utilizados é o *quadcopter*. Com um comportamento similar aos helicópteros, tem a capacidade de descolar verticalmente e permanecer estaticamente no ar, deslocando-se em qualquer direção sem a necessidade de realizar um movimento semelhante ao dos aviões tradicionais. Como tal, permite uma condução muito mais ágil para o piloto, libertando-o de movimentos constrangedores num curto espaço de manobra, permitindo também, devido à sua estabilidade, uma observação mais detalhada sobre objetos ou situações a analisar.

Com o surgimento de sensores estabilizadores *on-board*, como sucede no *quadcopter* desenvolvido pela empresa *Parrot* denominado de *AR.Drone* [8], apresentado na Figura 1.2, é possível direcionar a investigação para o desenvolvimento de aplicações que amplifiquem a sua capacidade de observação.

Figura 1.2 *Parrot AR.Drone 1.0* [8]

Em situações atmosféricas com vento forte, devido ao seu peso diminuto, o *drone* apresenta algumas dificuldades de controlo e de estabilização. Essa circunstância, aliada às limitações dos sensores embutidos e aos possíveis movimentos bruscos realizados pelo *drone*, transforma o controlo de navegação, um verdadeiro desafio.

Face às dificuldades apresentadas, a introdução de um sistema de navegação via GPS considera-se imprescindível, permitindo que se realize uma navegação de forma autónoma a uma velocidade de execução (na ordem dos milissegundos) não alcançável pelo ser humano.

Mesmo que este seja tripulado por um piloto com várias horas de voo, sistemas de navegação autónomos proporcionam um auxílio considerável, reduzindo a complexidade. A combinação da tecnologia GPS e dos sensores *Inertial Measurement Units* (IMUs) originam, assim, um aumento na capacidade de estabilização por parte dos *drones*.

A introdução da navegação via GPS, veio proporcionar não só a localização geográfica da aeronave, como também a possibilidade de inclusão de *waypoints* (WP), gerando uma rota pré-definida, simplificando a tarefa do piloto em situações em que não seja essencial a existência de controlo absoluto da parte do piloto. Os WPs [9] não são mais do que um conjunto de coordenadas ou pequenas regiões que fornecem a informação de um ponto físico no espaço. A partir destes, é possível que o *drone* prossiga numa rota pré-definida.

Neste panorama, decidiu-se desenvolver um sistema de navegação autónomo para *drones* tendo como base a receção de coordenadas GPS.

1.2 Objetivos e Contribuições

O trabalho desenvolvido nesta dissertação irá focar-se na implementação de um sistema de navegação autónomo que permita a um *quadcopter* realizar um rota pré-definida e regressar ao ponto de partida (também frequentemente designado por '*home'*).

O sistema de navegação autónomo baseia-se na receção de mensagens GPS, descodificação e análise das coordenadas, passando pela implementação de sistemas de segurança prevenindo possíveis falhas de navegação e concluindo com o envio de comandos de controlo para o *quadcopter*.

Para atingir o pretendido, a dissertação apresenta como objetivos:

- Controlo e navegação do *quadcopter Parrot AR.Drone 2.0*;
- Receção e descodificação de mensagens GPS através de comunicação série para *arduino*;
- Realização de algoritmos de segurança;
- Algoritmos de navegação autónoma e controlo gerados e transmitidos entre recetor GPS, *arduino* e *drone*, através de comunicação série;
- Interligação física entre os vários blocos de *hardware*;
- Testes de fiabilidade.

Como tal, o sistema irá estar dividido entre três segmentos fundamentais:

- **Segmento base:** Toda a estrutura de *software* necessária para iniciar, monitorizar e controlar o voo;
- **Segmento aéreo:** O *quadcopter* e todo o *hardware* em combinação com o *software* necessário para realizar a navegação autónoma e controlo de voo;
- **Segmento de comunicações:** Ligações e protocolos de comunicação entre o segmento base e o *quadcopter* (segmento aéreo).

Para alcançar os objetivos pretendidos é necessário interligar funcionalmente o segmento aéreo com o segmento base, tendo como interface, o segmento de comunicações.

Nesta dissertação serão apresentadas as arquiteturas e as soluções desenvolvidas tendo em vista o objetivo final, bem como os aspetos essenciais na implementação do sistema, finalizando com uma discussão sobre os resultados obtidos.

1.3 Estrutura do Documento

A dissertação encontra-se organizada em cinco capítulos distintos (para além do capítulo introdutório) de forma a simplificar a leitura e compreensão da informação que se pretende transmitir ao leitor.

No Capítulo 2, Conceitos Fundamentais do Sistema de Localização, é iniciado com um breve historial do estado da arte dos sistemas de navegação existentes, prosseguindo com uma descrição das técnicas de posicionamento desenvolvidas ao longo da evolução tecnológica. Finaliza-se o capítulo descrevendo o *modus operandi* do sistema de localização GPS.

No Capítulo 3, Conceitos Fundamentais do *Drone*, pretende-se descrever de forma sucinta, os conceitos gerais de controlo do *drone*. Será descrito de uma forma mais detalhada todo o *hardware* embutido no seu interior tal como os protocolos de comunicação e controlo, que servem de base para o desenvolvimento do sistema em questão.

No Capítulo 4, Plataforma de Navegação, inicializa-se com uma descrição sucinta dos tipos de mensagens recebidas pelo recetor de GPS.

Serão também descritos detalhadamente os aspetos técnicos do *hardware* necessário para coletar as mensagens provenientes dos satélites, descodificar as coordenadas recebidas e determinar a rota a seguir, enviando finalmente os controlos necessários para a concretização.

No Capítulo 5, Implementação do Sistema, são apresentados detalhadamente os algoritmos de controlo e de suporte na navegação autónoma.

É ainda descrito o acondicionamento do *hardware* necessário para a implementação do sistema. De seguida expor-se-ão os testes realizados ao sistema desenvolvido, seguidos de um breve comentário alusivo aos mesmos.

No Capítulo 6, Conclusões e Trabalho Futuro, far-se-á um resumo geral do trabalho realizado e apresentado e, também, uma apreciação final acerca dos resultados obtidos. Por fim, apresentar-se-ão aspetos a serem melhorados, tais como soluções que poderão ser desenvolvidas num futuro próximo.

Além dos capítulos apresentados, foram ainda elaborados anexos onde se poderão encontrar informações complementares ao documento:

No Anexo A, Comandos referentes à API do *AR.Drone 2.0*, são detalhados os comandos AT reconhecidos pela API do *drone*. Far-se-á de igual modo, a descrição das configurações iniciais necessárias para a realização de um voo controlado.

No Anexo B, Pinos de Entrada do recetor GPS, são descritas as propriedades de todos os pinos presentes no recetor.

No Anexo C, GPS Ligações físicas Recetor GPS ⇔ *Arduino* ⇔ *Drone*, são apresentadas as ligações físicas do *hardware* inerente ao sistema de navegação.

Capítulo 2

Conceitos Fundamentais de Navegação

Neste capítulo é realizada uma abordagem histórica sobre as primeiras técnicas de navegação existentes e o seu desenvolvimento ao longo de uma era de constante evolução na comunicação tendo como base o espaço, através de uma descrição cronológica da evolução tecnológica.

Terminar-se-á com a descrição de forma detalhada do princípio de funcionamento do sistema de posicionamento mais conhecido e utilizado no planeta, o GPS.

2.1 Sistema de Localização

Desde os primórdios da civilização, o ser humano tem olhado para o céu como sendo uma fonte de inspiração com um grande potencial. Alguns desses homens tornaram-se especialistas em extrair conhecimento através das estrelas e a desenvolver processos que permitissem absorver informação baseando-se no seu posicionamento. Hoje em dia, é conhecido que o alinhamento de estruturas edificadas mundialmente conhecidas (ex., pirâmides do Egipto) foi definido através de observações celestiais e as próprias estruturas são utilizadas na determinação de acontecimentos como o equinócio primaveril. A bibliografia utilizada no sub-capítulo resumiu-se essencialmente à referência [10].

2.1.1 Desenvolvimento de Técnicas de Posicionamento

A primeira técnica de posicionamento utilizada foi a triangulação, consistindo num processo de determinação da localização de um ponto geográfico, através da determinação dos ângulos entre pontos conhecidos em ambas as extremidades de uma base fixa e de comprimento fixo. No entanto, a técnica era limitada não sendo possível realizar medições sem que os pontos nas extremidades tivessem em linha de vista.

Com a evolução da investigação na astronomia, no dia 4 de Outubro de 1957 é enviado para o espaço o primeiro satélite artificial, o satélite russo *Sputnik*, contribuindo colossalmente para a evolução da ligação dos vários pontos de referência da Terra.

2.1.2 Posicionamento Baseado em Satélites

O posicionamento baseado em satélites representa a determinação da localização de pontos geográficos, através dos satélites. O princípio de funcionamento passa por fornecer ao utilizador, a partir de recetores eletrónicos, a capacidade de determinar a sua localização, expressa em latitude, longitude e altitude, utilizando sinais de tempo transmitidos pelos rádios do satélite. É possível calcular o tempo atual com grande precisão, o que possibilita efetuar uma sincronização bastante precisa. O sistema de navegação global baseado em satélites é denominado de *Global Navigation Satellite System* (GNSS).

Na Figura 2.1 é possível perceber o princípio do posicionamento de um recetor, calculando através da equação 2.1, a distância do recetor ao satélite $\boldsymbol{\varrho}$, correlacionada entre a distância do recetor ao centro da Terra $\boldsymbol{\varrho_r}$ e a distância do centro da Terra até ao satélite $\boldsymbol{\varrho^s}$.

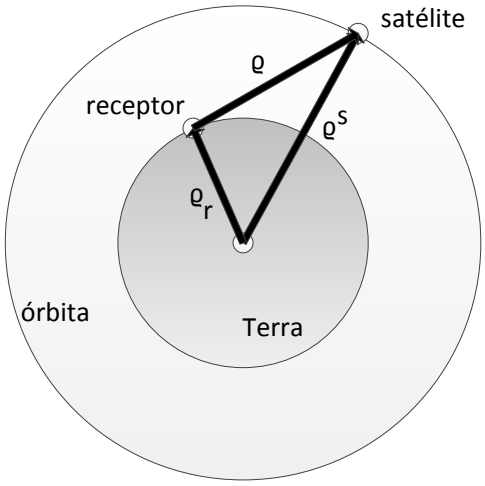

Figura 2.1 Princípio do posicionamento baseado em satélites

Em que,

$$\varrho = \|\varrho^s - \varrho_r\| \tag{2.1}$$

No entanto, os recetores modernos utilizam uma técnica ligeiramente diferente, consistindo na utilização de um relógio configurado com aproximadamente o mesmo tempo do sistema. Desta forma, o relógio do recetor sofre de um ligeiro *offset* relativamente ao sistema de tempo real gerado pelos satélites e devido a este *offset*, a distância medida até ao satélite é ligeiramente díspar da distância geométrica.

Como consequência, a distância medida é denominada de pseudo-distância **R** sendo que esta representa a distância geométrica adicionando a sua retificação $\boldsymbol{\Delta_{\varrho}}$, resultante do erro de relógio sistemático **δ**, dando origem à equação 3.2:

$$R = \varrho + \Delta_{\varrho} = \varrho + c\delta \tag{2.2}$$

Em que c representa a velocidade da luz.

2.1.3 Sistemas Baseados no Posicionamento em Satélites

Na era antecedente aos atuais sistemas de posicionamento baseados em satélites, surgiu o *Navy National Satellite System* (NNSS) também denominado por *Transit*. Na Figura 2.2 é possível visualizar o primeiro protótipo a ser preparado para o seu lançamento. A 13 de Abril de 1960 foi lançado com sucesso para a órbita auxiliando o serviço naval dos Estados Unidos em 1964. O seu principal objetivo era determinar as coordenadas dos navios no alto mar, servindo ainda de suporte nas aterragens de aviões militares.

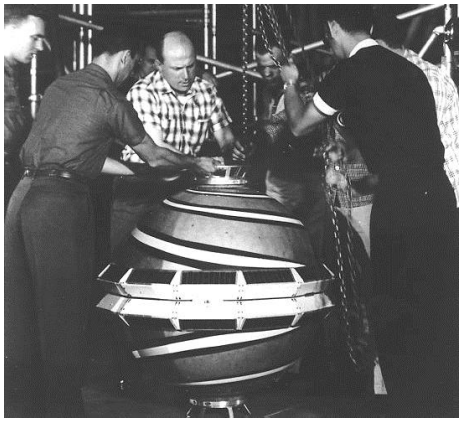

Figura 2.2 Primeiro protótipo do *Transit 1A* a ser preparado para o lançamento [11]

Nos anos 70, a União Soviética deu continuação à produção dos seus próprios sistemas de navegação. O funcionamento do sistema de navegação *Tsikada* é semelhante ao funcionamento do *Transit*, tanto na transmissão de informação, como na sua capacidade precisa de receção. Composto por duas constelações, o primeiro satélite foi lançado em 1974. Contrariamente ao *Transit*, este ainda se encontra operacional.

Os sistemas antigos, no entanto, apresentavam algumas lacunas. O principal problema estava no grande tempo morto entre as passagens dos satélites. A título exemplificativo, um satélite estava disponível a cada noventa minutos, obrigando os utilizadores a interpolar a sua posição entre as captações ou entre as passagens. O segundo problema deveu-se à baixa precisão de navegação, nomeadamente do parâmetro da altura.

Nos sistemas de navegação atuais, destaca-se o sistema *Navigation Sattelite with Time and Ranging Global Positioning System* (NAVSTAR GPS) que foi desenvolvido para fins militares nos Estados Unidos fazendo face às limitações do sistema *Transit*. Desenvolvido em 1973, era constituído originalmente por 24 satélites e tornou-se completamente funcional em 1995. Inspirados no sistema de navegação GPS, outros sistemas entraram em funcionamento.

Seguiu-se o satélite russo *Global Navigation Satellite System* (GLONASS) cujo exemplar é observado na Figura 2.3, tendo sido desenvolvido em 1976 para fins militares, no entanto, apenas em 2010 o GLONASS viria a atingir a cobertura de 100% do território Russo, tendo essa demora sido provocada pelo desmembramento da União Soviética e consequente tomada de posse do satélite pela Rússia. Em Outubro de 2011, atingiu finalmente a cobertura total do planeta. Difere do GPS em termos do segmento de controlo, segmento de espaço e da estrutura do sinal.

Figura 2.3 Satélite GLONASS [12]

Mais recentemente, a União Europeia uniu-se para desenvolver o seu próprio GNSS, junto da *European Space Agency* (ESA), respondendo assim aos desafios das tecnologias de informação e da necessidade de contribuir para a evolução da navegação baseada em satélites. Contrariamente aos já existentes GLONASS e GPS, a ESA desenvolveu um sistema de navegação para uso civil, não restringindo assim o sinal aos civis, como tal acontece com o GPS. Foi denominado de *Galileo* em homenagem ao cientista e astrónomo italiano, *Galileu Galilei*. Um dos seus exemplares pode ser observado na Figura 2.4.

Figura 2.4 Satélite *Galileo* [13]

É espectável que em 2019 esteja completamente funcional com uma cobertura global.

Também no continente asiático, nomeadamente a República Popular da China tomou a decisão de desenvolver o seu próprio sistema de navegação, denominado de *BeiDou Satellite Navigation System* (BDS). Tal sistema está a ser desenvolvido desde Janeiro de 2013 e é espectável que venha a ter ao seu dispor 35 satélites. Desde 2012, que estão disponíveis e operacionais, com cobertura na região do pacífico, 10 satélites, planeando-se que, em 2020, tenham uma cobertura à escala global.

2.2 *Global Position System*

Tal como anteriormente referido, o GPS é um sistema de navegação baseado em satélites desenvolvido pelo *Deparment of Defense* (DoD) dos Estados Unidos da América, no início dos anos 70 [14].

Inicialmente o GPS foi desenvolvido como um sistema para colmatar as necessidades do sistema militar americano. No entanto, foi mais tarde disponibilizado para os civis, apesar de algumas restrições impostas.

O GPS é um sistema com capacidade para fornecer dados de forma contínua da posição e do tempo de um recetor, em qualquer lugar ou sobre qualquer condição atmosférica. Devido a ser utilizado por um número infinitesimal de recetores e por motivos de segurança, o GPS é um sistema unidirecional, não tendo o utilizador a capacidade de enviar informação para os satélites apenas sendo capaz de receber o seu sinal de localização.

O GPS comporta, numericamente, uma constelação de 24 satélites, tal como é reportado na Figura 2.5. Esta constelação foi completada em Julho de 1993. Sob forma de garantir constantemente uma cobertura global, os satélites GPS circulam agrupados equitativamente em 6 planos orbitais distintos. Com esta geometria, quatro (o mínimo essencial para facultar informação precisa acerca da posição do recetor) satélites estão visíveis em qualquer ponto do planeta.

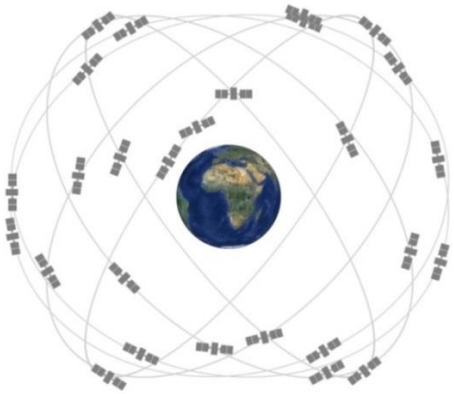

Figura 2.5 Constelação dos 24 satélites GPS dispostos em 6 órbitas distintas [15]

As órbitas do GPS em forma elipsoidal possuem uma inclinação de 55º relativamente ao equador e encontram-se a uma altura de aproximadamente 26,600 km da Terra, sendo o período orbital de um satélite é sensivelmente de 12 horas.

O sistema GPS foi oficialmente declarado como completamente funcional a 17 de Julho de 1995, assegurando a disponibilidade de pelo menos 24 satélites operacionais. No entanto, desde o anúncio, o número de satélites não estagnou, fixando-se atualmente em 32 satélites ativos [16].

2.2.1 Segmentos

A estrutura do GPS consiste em três segmentos distintos:

- Segmento do espaço;
- Segmento de controlo;
- Segmento do utilizador.

A disposição e a interligação dos segmentos encontram-se descritas na Figura 2.6.

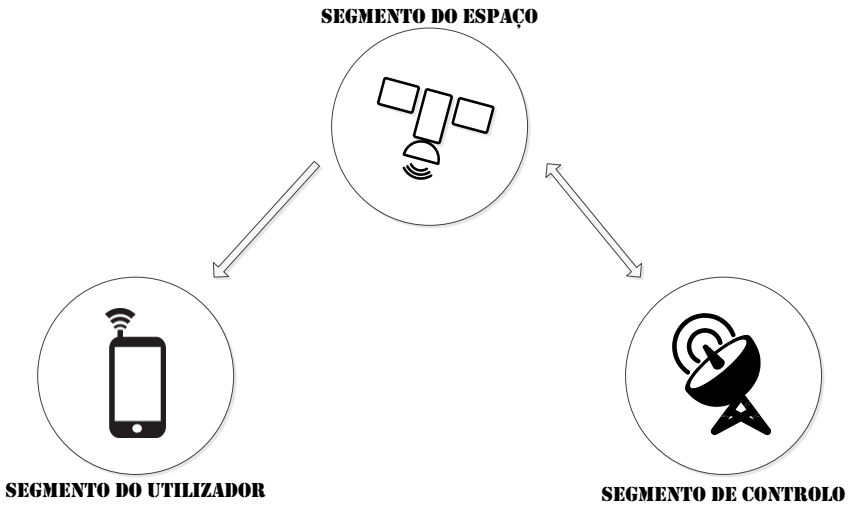

Figura 2.6 Segmentos do GPS

2.2.1.1 Segmento do Espaço

Tal como já referido, o segmento do espaço é constituído por uma constelação atualmente de 32 satélites. A inclusão de satélites complementares traduziu-se num aumento da precisão do recetor GPS. Com o aumento do número de satélites disponíveis, a constelação deixou de ter uma distribuição uniforme, o que veio traduzir-se num aumento da fiabilidade e a disponibilidade relativamente ao sistema uniforme, numa eventual situação de vários satélites falharem [17].

Na configuração padrão, cada satélite transmite um sinal, que terá as seguintes componentes:

- Duas frequências:
    - L1 a 1575.42 GHz
    - L2 a 1227.67 MHz
- Dois códigos digitais:
    - *Coarse Acquisition code* (*C/A code*)
    - *Precision Code* (*P-code*) também designado por PRN. No entanto, como medida preventiva contra o possível *hacking* do sinal militar (*spoofing*), decidiu-se encriptar o código, passando a denominar-se de *P(Y)-code*.
- Uma mensagem de navegação.

O comprimento de onda das portadoras é de aproximadamente 19 cm e 24 cm, respetivamente, resultante da relação entre a frequência da portadora e a velocidade da luz no espaço.

No entanto com a modernização realizada, imposta pela introdução de novos satélites [18] e pela necessidade de realizar uma evolução nos sinais transmitidos para utilizadores civis, foram introduzidos dois novos sinais, o L2C e o L5.

Gerações

Ao longo da implementação do sistema de GPS, foram sendo lançados para o espaço várias gerações de satélites [14].

A primeira geração, designada de *Block I*, surgiu com o lançamento de 11 satélites que começaram a ser enviados para o espaço no dia 22 de Fevereiro de 1978, sendo o último satélite da geração enviado no dia 9 de Outubro de 1985.

O seu principal objetivo era servir de base para o desenvolvimento do sistema de navegação. Apesar de um tempo de vida previsto de 4 anos, muitos deles permaneceram em funcionamento mais de 10 anos, tendo o último satélite sido desativado a 18 de Novembro de 1995.

A Segunda geração foi denominada de *Block II* e *Block IIA*. A evolução do *Block IIA* surgiu com uma maior capacidade de armazenamento de mensagens de dados de navegação de 14 para 118 dias, sem qualquer necessidade de contacto com o segmento de controlo. Foram lançados para o espaço 28 satélites desde Fevereiro de 1989 até Novembro de 1997, estando ainda operacionais 5 satélites.

Na terceira geração, *Block IIR*, os satélites começaram a ser enviados para o espaço a partir do dia 23 de Julho de 1997, com o último lançamento a ser efetuado em Agosto de 2009.

Com total compatibilidade com a geração anterior, os satélites têm a capacidade de medir a distância entre eles e de transmitir dados para outros satélites e para o segmento de controlo. Tal como a geração anterior, os satélites utilizam as mesmas frequências e os mesmos códigos para a transmissão de dados. É composta por 20 satélites, estando, no entanto, apenas 12 operacionais.

A inclusão da quarta geração [19], *Block IIR-M*, veio reforçar a constelação de satélites GPS, com novos satélites e mais modernizados. Estes vieram introduzir um novo sinal militar e reforçar a robustez dos sinais civis L2C. Composto por 8 novos satélites, o primeiro foi lançado em Setembro de 2005, sendo concluído o processo com o último lançamento a ser concretizado em Agosto de 2009. Dos 8 satélites enviados para o espaço, 7 continuam ainda totalmente operacionais.

A quinta e última geração, *Block IIF*, veio expandir a capacidade da série *IIR-M*, com a adição de um terceiro sinal civil (L5) exclusivamente para transportes em situações de emergência.

Surgem ainda com uma precisão aperfeiçoada, bem como um sinal de transmissão mais forte. Composto por 12 satélites, o primeiro lançado efetuou-se a 28 de Maio de 2010, tendo o sétimo e último lançamento sido realizado a 2 de Agosto de 2014. Foi então lançado como previsto, outro satélite a 29 de Outubro de 2014, porém ainda não se encontra em atividade.

Está prevista uma nova geração, *GPS III*, que introduzirá sinais ainda mais fortes com uma maior fiabilidade, precisão e integridade, o que garantirá uma maior capacidade de navegação. Apresentará o quarto sinal civil (L1C), bem como um sistema de alerta de emergência para procura e resgate. Prevê-se que os primeiros satélites estejam disponíveis para lançamento a partir de Janeiro de 2016. Na Figura 2.7 é possível visualizar-se a evolução dos satélites *GPS* ao longo do tempo.

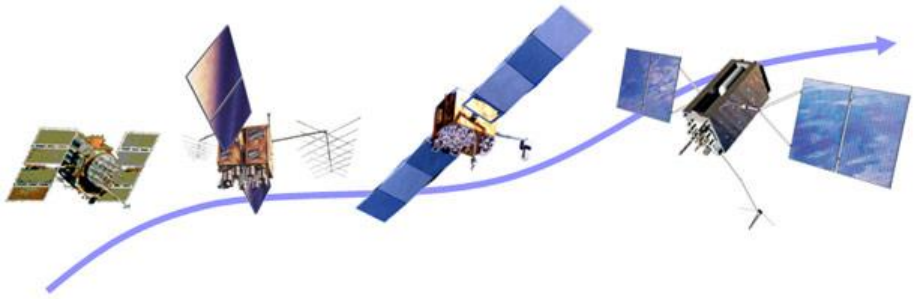

Figura 2.7 Evolução dos satélites GPS (da esquerda para a direita): *Block IIA, Block IIR(M), Block IIF e GPS III* [20]

2.2.1.2 Segmento de Controlo

O segmento de controlo, também frequentemente designado como segmento de base, consiste numa rede global de vigilância dos satélites da constelação, presente na Figura 2.8.

Este é composto maioritariamente por quatro subsistemas [21]:

- Estação de Controlo Principal (MCS - *Master Control Station*);
- Estações-Monitor à escala global (MS - *Monitor Stations*);
- Sistema Alternativo da Estação Principal (AMCS - *Alternate Master Control Station*);
- Antenas Terrestres (GA - *Ground Antennas*).

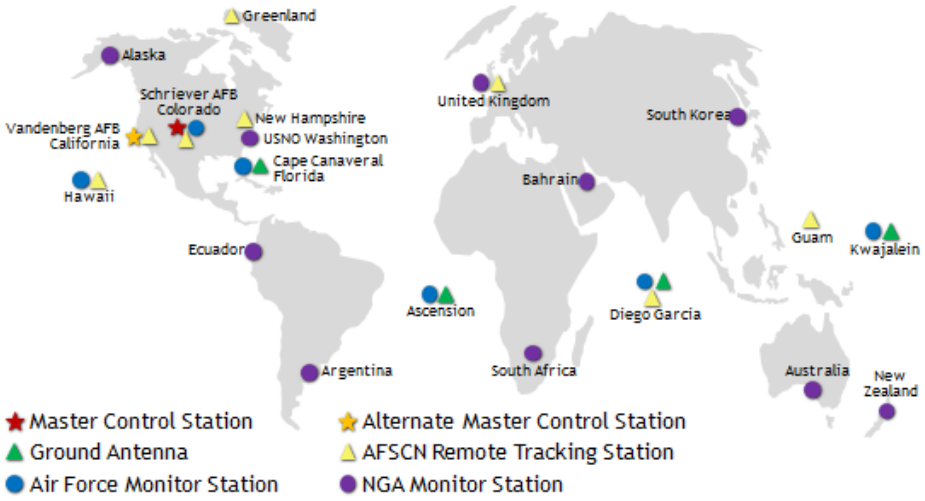

Figura 2.8 Segmento de Controlo [21]

Este segmento permite que se realize o acompanhamento em tempo real dos satélites, se monitorizem as transmissões/receções de dados entre os satélites e os recetores e se enviem dados para a constelação.

O segmento de controlo operacional atual é composto por uma MCS situada nos Estados Unidos, mais concretamente no Colorado, contando com o suporte de 12 antenas terrestres espalhadas pela Terra e ainda 16 estações-monitor. Conta ainda com o sistema de AMCS, estrategicamente colocado na Califórnia, Estados Unidos.

Estação de Controlo Principal

A estação de controlo principal executa as funções principais do segmento de controlo, gerando os comandos e realizando o controlo da constelação, assegurando a precisão dos satélites, bem como do seu estado físico e correto funcionamento.

Estações-Monitor

A missão das estações é a de acompanhar o movimento dos satélites GPS, bem como analisar a cobertura terrestre do satélite, transmitindo as imagens visualizadas para a MCS. As estações utilizam recetores GPS sofisticados, sendo controladas a partir da MCS.

Antenas Terrestres

As antenas terrestres têm como função, comunicar com os satélites GPS com a finalidade de enviar comandos ou controlos.

2.2.1.3 Segmento de Utilizador

Este é o segmento que é composto por milhões de dispositivos com recetores de sinal GPS, bem como centenas de milhares de recetores militares dos Estados Unidos e respetivos aliados.

Na generalidade, os recetores são compostos por uma antena sintonizada à frequência transmitida pelos satélites, processadores da informação recebida e um relógio altamente estável, frequentemente composto por um oscilador de cristal.

Aplicações

Hoje em dia qualquer meio de transporte aéreo ou marítimo tem incorporado um recetor GPS. A inclusão em meios de transportes terrestres começa a ser cada vez mais uma realidade e não apenas em situações pontuais. A diversidade de recetores GPS fizeram com que se tornasse uma ferramenta indispensável numa vasta gama de possíveis cenários:

- Astronomia;
- Automação veicular permitindo que os veículos sejam localizados ou inclusive para condução autónoma;
- Cartografia;
- *Smartphones*;
- Serviços de emergência médica;
- Roteiros turísticos;
- Robótica;
- Desporto;
- Etc.

Na Figura 2.9, é mostrado um exemplar comum de recetor GPS aplicado na orientação veícular.

Figura 2.9 Exemplo de recetor GPS [22]

Localização

Na receção de dados enviados pelo satélite, o recetor determina a sua posição através do tempo preciso em que o satélite enviou a informação, sob a forma de:

- Tempo em que a mensagem fora enviada;
- Posição do satélite no momento da transmissão da mensagem.

Partindo da mensagem enviada, o satélite determina o tempo de transmissão, calculando depois a distância tendo em consideração a velocidade da luz. São efetuadas várias leituras de um agregado de posições, definindo uma área de abrangência (plano) e portanto, o recetor estará posicionado num ponto da extremidade do plano.

Teoricamente, o princípio base da localização requer três satélites com posições distintas, tendo cada um definido o seu plano de cobertura. Neste caso o recetor deverá estar posicionado na interseção dos três planos abrangentes [23]. Cada satélite tem o seu raio de cobertura com o valor da distância do satélite ao recetor, tal como descrito na Figura 2.10. No entanto, devido a não estar presente no recetor um relógio atómico, este não possui uma sincronização precisa com os satélites, sendo necessária a inclusão de um quarto satélite, evitando que o *offset* do relógio recetor tenha que ser considerado, possibilitando ao recetor estar equipado com um relógio comum, permitindo que o preço a que este é comercializado, seja um valor razoável e praticável.

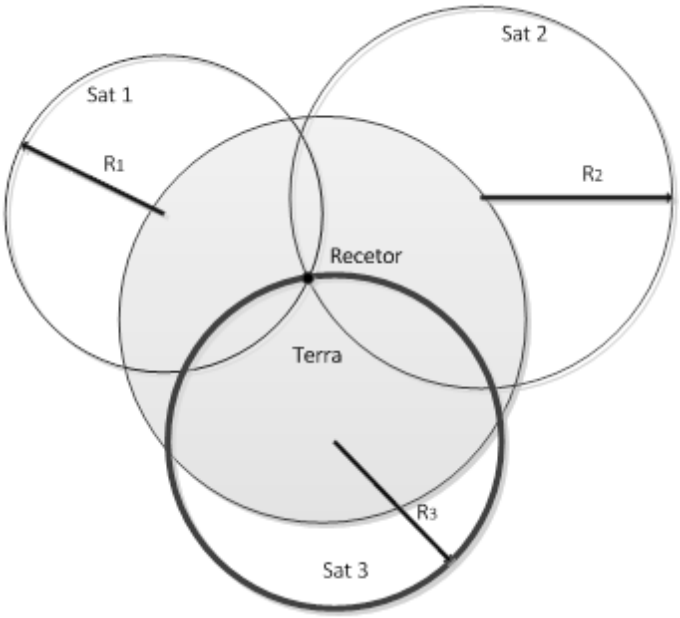

Figura 2.10 Localização com base na técnica de triangulação

O lançamento do projeto *European Geoestationary Navigation Overlay Service* [24] (EGNOS) corresponde a um sistema desenvolvido com o objetivo de aumentar a precisão dos sinais recebidos pelos satélites que sobrevoam a Europa. O sistema é baseado em *transponders* introduzidos em três satélites geoestacionários sobre a Europa, interligados com 40 estações-base e 4 centros de controlo. Com este sistema foi possível uma redução do erro de localização de 20 para 2 metros. Contudo, acredita-se que com a evolução do projeto europeu *Galileo*, o erro poderá ser reduzido para poucos centímetros.

Capítulo 3

Conceitos Fundamentais do *Drone*

Hoje em dia, verifica-se um crescimento exponencial de investimento nos *drones*. Assim, têm sido realizados vários estudos sobre a utilização daqueles e sobre que tipo de tecnologias devem ser introduzidas, por forma a otimizar os voos realizados.

Numa altura em que se exigem produtos inovadores com a melhor tecnologia embutida, a empresa *Parrot*[1] surge como uma das mais inovadoras na conceção de *quadcopters*, introduzindo uma gama de sensores que permitem a execução de voos suaves, sem que o piloto seja obrigado a assumir um controlo total sobre o veículo.

Neste capítulo serão apresentados os conceitos fundamentais para uma melhor compreensão do trabalho desenvolvido.

Introduzir-se-á uma visão geral acerca do *drone* que serve de plataforma para a realização do trabalho proposto.

Concluir-se-á com uma exposição detalhada sobre as formas de comunicação disponíveis, os comandos necessários para controlar os movimentos, tal como a transmissão e receção de dados necessários para tal.

Todo o capítulo é detalhado, tendo como base o manual de desenvolvimento do *AR. Drone* [25].

3.1 Parrot AR.Drone

Em 2004, a empresa Parrot deu início ao projeto denominado de *AR.Drone*, com a intenção de desenvolver um MAV que se tornasse mundialmente popular, com a capacidade de penetração no mercado do entretenimento pessoal bem como no dos videojogos.

[1] http://ardrone2.parrot.com/

O projeto foi apresentado publicamente em 2010 no *Consumer Eletronic Show* (CES). A 18 de Agosto de 2010, o *AR.Drone* é oficialmente lançado no mercado. Uma das principais características do *quadcopter* apresentado foi a plataforma de estabilização aérea, controlada autonomamente através de uma interface gráfica desenvolvida para *iPhone, iPad* ou *iTouch*. Foi disponibilizado para vários países em lojas de retalhe e na *Apple store* a um preço de cerca de 300€.

A estrutura física do *AR.Drone* [26] é essencialmente composta por *nylon* e fibra de carbono. O casco de proteção superior é removível, estando disponível um casco para cenários exteriores e outro para interiores.

O casco de interior é constituído por espuma, tendo uma estrutura de quatro círculos protetores protegendo as hélices de pequenos embates em paredes ou outros objetos. O casco exterior é também ele composto por espuma, no entanto de menor dimensão, o que permite o aumento de mobilidade. A empresa desenvolveu um *Micro-Electro-Mechanical System* (MEMS) de navegação inercial composto por um acelerómetro de três-eixos, um giroscópio de dois-eixos e um giroscópio de um-eixo para o movimento de rotação, permitindo melhorar o processo de estabilização.

No interior da estrutura protetora inferior existe ainda um sensor de ultrassons e uma câmara, permitindo auxiliar a interface utilizada pelos pilotos, possibilitando também que o voo se torne mais simples de realizar.

Em 2012 no CES Las Vegas [27], eis que é revelado o seu sucessor, *AR.Drone 2.0*. Mantendo um *design* muito semelhante ao anterior, as evoluções concentraram-se nas suas funcionalidades, juntamente com o desenvolvimento de um amplo sistema de suporte aos pilotos.

Os componentes da *motherboard* no *AR.Drone 2.0* foram atualizados para versões mais recentes e com maior precisão [28] por forma a aumentar a fiabilidade do *drone*.

Também ele comercializado com um preço-base de cerca de 300€, apesar de designado como um brinquedo de alta tecnologia, rapidamente captou a atenção de várias universidades e institutos de investigação.

Apresentado na Figura 3.1, tem as dimensões [29] de 52.5 x 51.5 cm com o casco interior e 45 x 29 cm com o casco exterior, possui quatro motores munidos de pás de 20 cm de diâmetro, interligados através de uma armação de fibra de carbono cruzada. O peso do *drone* varia entre os 360g com o casco exterior e os 420g com o casco interior.

Figura 3.1 *AR.Drone 2.0* com casco interior (esquerda) e casco exterior (direita) [30]

3.2 Movimentos Básicos

O *quadcopter* é controlado através dos quatro motores existentes, gerando assim um movimento rotacional, vertical, horizontal ou combinado. Assim, este pode ser controlado numa perspetiva tridimensional somente ajustando a velocidade dos motores individualmente, consoante o movimento pretendido. Na Figura 3.2 são descritas as variáveis dos movimentos básicos realizados.

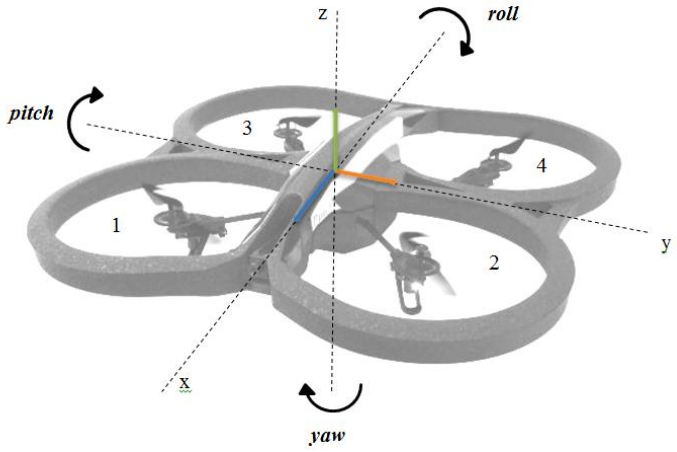

Figura 3.2 Variáveis dos movimentos básicos do *quadcopter* [31] (imagem editada)

A variação da velocidade de rotação dos motores de forma independente implicará a alteração dos valores das quatro variáveis (*pitch*, *yaw*, *roll*, *gaz* e os respetivos ângulos de *pitch* e *roll*), gerando desta forma os distintos movimentos que executados pelo *drone*. São apresentadas na Figura 3.3, consoante a velocidade aplicada nos motores, os quatro movimentos distintos.

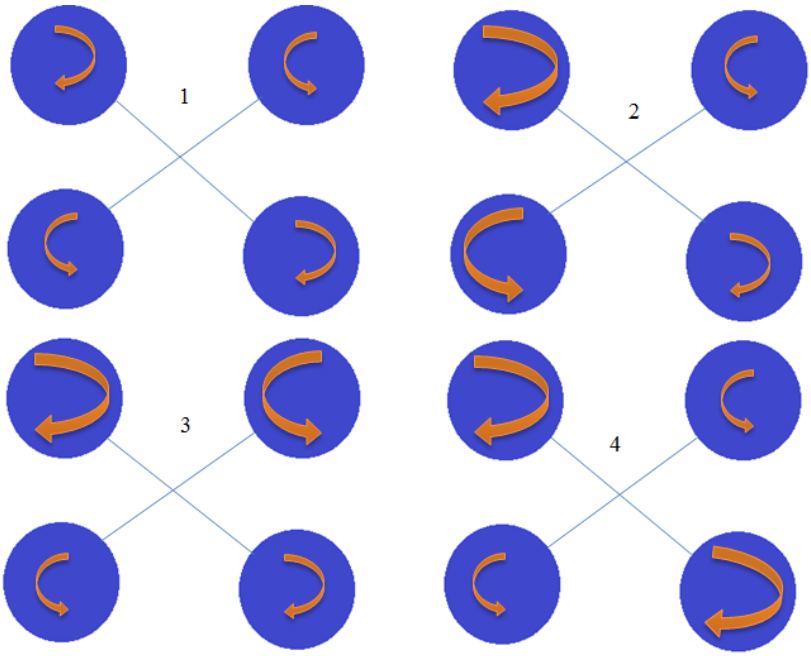

Figura 3.3 Movimentos básicos: 1- Aceleração Vertical; 2- Movimento lateral; 3- Movimento frontal; 4- Rotação

Resumindo, as ações básicas de um *quadcopter* são:

- Movimento vertical (*gaz*): obtido com o aumento ou diminuição da velocidade equitativamente nos quatro motores;
- Movimento lateral (*roll*): obtido pelo aumento da velocidade de dois motores laterais (1 e 3 provocando uma deslocação para a direita; 2 e 4 provocando uma deslocação para a esquerda), enquanto a velocidade dos motores na posição oposta é reduzida, resultando assim numa inclinação que dá origem ao movimento lateral;
- Movimento frontal (*pitch*): obtido pela diminuição (ou aumento) da velocidade de ambos os motores dianteiros (1 e 2) e aumento (ou diminuição) da velocidade dos motores traseiros (3 e 4) fazendo baixar (ou elevar) da parte frontal do *drone*, dando origem a um movimento para a frente (ou para trás).
- Rotação (*yaw*): obtida com o aumento da aceleração dos motores 1 e 4, enquanto a aceleração dos motores 2 e 3 é decrementada (ou vice-versa), obtendo um movimento no sentido dos ponteiros do relógio (ou no sentido inverso aos ponteiros do relógio), sem que seja necessária a mudança da posição ou altura do *quadcopter*;

3.3 Hardware

Com quatro hélices controladas por quatro motores (28,500 rpm @ 14.5W) alimentados por uma bateria de lítio de 1000 mAh @ 11.1V, o *drone* proporciona um voo de cerca de dez minutos de duração.

No interior da estrutura protetora inferior [29] encontra-se na *motherboard* um microprocessador ARM Cortex A8 @ 1GHz, que corre sobre um SO *Linux*, com 1Gb de RAM DDR @ 200MHz, exibido na Figura 3.4. É também disponibilizada uma porta USB 2.0 que permite a conexão de periféricos externos para armazenamento de vídeo, imagem ou dados. É fornecida uma *Application Progamming Interface* (API) para comunicação entre o *drone* e o piloto utilizando uma rede sem-fios. A API não só permite configurar os estados do *drone*, mas também fornece o acesso à informação processada pelos sensores e das imagens geradas pelas câmaras.

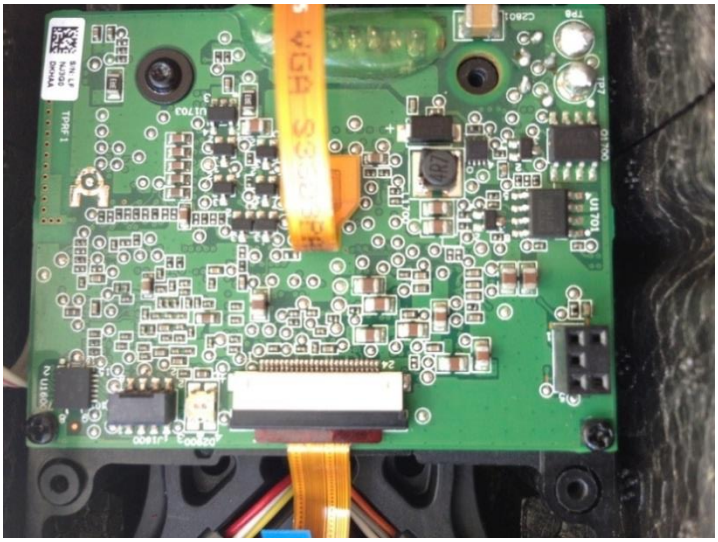

Figura 3.4 *AR.Drone 2.0 main board* [32]

Inicialmente o *drone* era apenas controlado via dispositivos *Apple* e mais tarde em 2011, por dispositivos com SO *Android*. No entanto, é também possível controlar o *drone* via SO *Linux* ou SO *Windows*, através de aplicações desenvolvidas por programadores.

3.3.1 Sensores

A evolução dos sensores a que foi sujeita a segunda versão do *AR.Drone* comprovou ser fundamental no auxílio a um controlo assistido mais fiável. A incorporação dos sensores com uma maior precisão, veio aumentar a capacidade de estabilização automática.

3.3.1.1 Unidade de Medição Inercial

O movimento realizado pelo *drone* é composto por seis parâmetros distintos denominados 6 *degrees of freedom* (DOF). O IMU [33] fornece ao microcontrolador os valores de *pitch*, *roll* e *yaw* e respetivos ângulos. Estes valores serão utilizados para efetuar a estabilização automática e para o controlo de movimentos.

O **acelerómetro** *on-board* do *drone* (com uma precisão de +/- 50mg) é composto por três eixos perpendiculares. Cada eixo é capaz de medir a aceleração na direção do mesmo.

O **giroscópio** presente na *motherboard* (com 6º de precisão) irá calcular a velocidade angular, baseando-se no princípio de momento angular. É composto por três eixos e cada um devolve o ângulo do *pitch* e *roll*.

3.3.1.2 Sensor de Pressão

Na versão mais recente do *Parrot AR.Drone*, foi incorporado um sensor de pressão com uma precisão de +/- 10 Pa. O *drone* passou a ter uma estabilidade de altura muito superior, nomeadamente a alturas superiores a 6 metros de altura que inviabilizem o ultrassom de realizar leituras precisas. Desta forma será possível alcançar alturas consideráveis mantendo a estabilidade e a posição no ar, sendo estas automaticamente corrigidas.

3.3.1.3 Sensor de Ultrassons

Tal como foi referido, o ultrassom à margem do sensor de pressão, fornece a leitura da altitude por forma a controlar a altura a que o *drone* se encontra do solo, bem como o controlo da aceleração vertical igualmente para fins de calibração da velocidade necessária para que este mantenha a sua posição relativamente ao solo. O sensor de ultrassons encontra-se embutido na parte inferior do *drone*, direcionado para o solo, por forma a medir a distância deste. Uma trama de ondas ultrassónicas é transmitida em direção ao solo, sendo refletido o som de volta ao sensor.

O sensor mede então o tempo **t** de retorno do eco, determinando a distância **d** do *drone* ao solo, pela fórmula:

$$d = \frac{c \times t}{2} \tag{2.1}$$

Onde **c** ≈ 343 m/s, representando a velocidade do som.

No entanto, como foi referido anteriormente, o alcance sofre de condicionantes fundamentais, no que toca à aeronáutica, tornando-o um sensor obsoleto.

A primeira e principal limitação é o pequeno alcance a que o sensor de ultrassons é preciso. No *AR.Drone 2.0* possui um alcance aceitável entre 20 cm até 6m, aproximadamente.

Outra limitação exposta pelo sensor deve-se ao facto de estes não serem capazes de obter uma informação exata da direção dos objetos. O som propaga-se em forma de cone, onde a abertura angular está na gama dos 20 a 40 graus. Devido a esta propriedade, o sensor obtém regiões com uma profundidade constante ao invés de obter pontos de profundidade discreta. Devido a tal facto, o ultrassom apenas consegue transmitir que existe um objeto a uma distância coberta pelo cone, não tendo a capacidade de determinar com exatidão a sua distância. Na Figura 3.5, é apresentado o princípio de funcionamento do ultrassom e também o módulo físico utilizado.

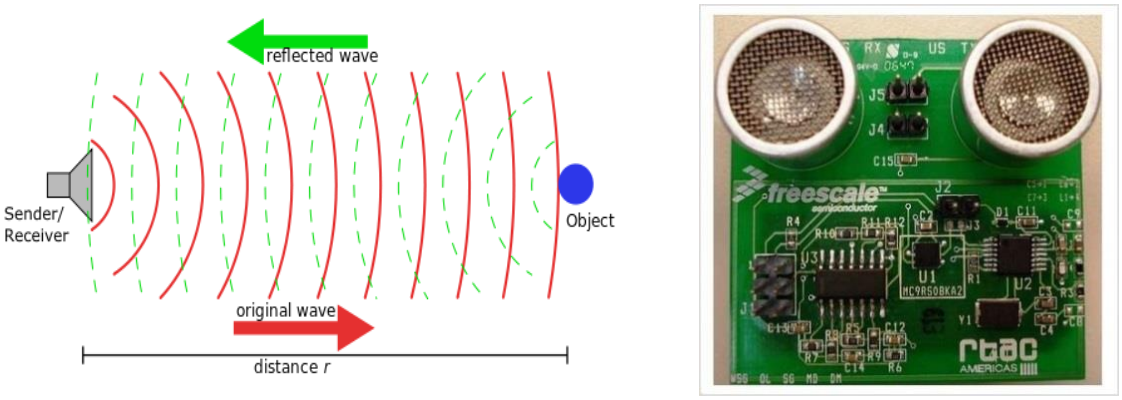

Figura 3.5 Funcionamento do sensor de ultrassom (esquerda) [34] e módulo físico (direita) [35]

O desempenho está também dependente das debilidades na orientação em determinadas superfícies. Quando uma onda é enviada na direção de um obstáculo com uma superfície paralela ao *drone*, uma grande quantidade da energia da onda é refletida perpendicularmente à superfície e irá ser coletada pelo sensor. Contudo, se a superfície do obstáculo for inclinada relativamente à posição do sensor, então apenas uma pequena quantidade de energia será refletida na sua direção, tornando-se um objeto indetetável.

Por fim, também a limitação da frequência de operação é um obstáculo. Antes de um ultrassom ser enviado, é necessário que a onda precedentemente enviada seja detetada pelo sensor.

3.3.1.4 Magnetómetro

O magnetómetro [36] desempenha um papel fundamental na nova versão do *AR.Drone*. Este é composto por três eixos com uma precisão de 6º.

O magnetómetro permite que o *drone* determine a sua orientação de forma precisa relativamente ao piloto, passando este a ser o ponto de referência. O piloto deixa de ter problemas com a orientação da câmara frontal, que irá controlar o movimento e a inclinação do *drone*.

3.3.1.5 Câmaras

O *AR.Drone 2.0* é equipado com duas câmaras CMOS [25], uma frontal e uma posicionada na zona inferior. Ambas suportam transmissão instantânea de vídeo a 15 FPS.

A câmara frontal tem uma resolução de 640 x 480 (VGA) e uma abertura de 93º de campo de visão. A aplicação mais relevante é a transmissão de vídeo em tempo real num ecrã de um *smartphone* ou *tablet* ou computador.

A câmara colocada na zona inferior do *drone* tem uma resolução de 176 x 144 (QCIF) com uma abertura de 63º de campo de visão. A frequência de transmissão de vídeo é de 60 FPS com o objetivo de reduzir as manchas produzidas pelo movimento e de melhorar a eficiência de algoritmos. Apesar da alta frequência, a transmissão é efetuada apenas a 15 FPS.

A câmara inferior tem ainda um papel fundamental na atuação *on-board* do *drone*. Esta é usada para a estabilização horizontal (*hovering* e *trimming*) e ainda permite estimar a velocidade instantânea.

3.4 Comunicação

O processo da comunicação entre uma aplicação (*Smartphone*, *Tablet* ou Computador) e o *drone* é realizada através do protocolo de transmissão de dados *User Data Protocol* (UDP) e sobre o protocolo de controlo de transmissão *Transmission Control Protocol* (TCP).

Ambos os protocolos estão presentes na camada de transporte (camada 4) do modelo representativo dos protocolos de Internet, *Open Systems Interconnection* (OSI) [37]. É ainda possível comunicar com o *drone* através de um microcontrolador via comunicação série ou ainda através de um computador a partir do protocolo de comunicação *telnet*. O protocolo pertence à camada de sessão (camada 5) do modelo OSI. Na Figura 3.6, é realizada uma breve descrição das camadas do modelo.

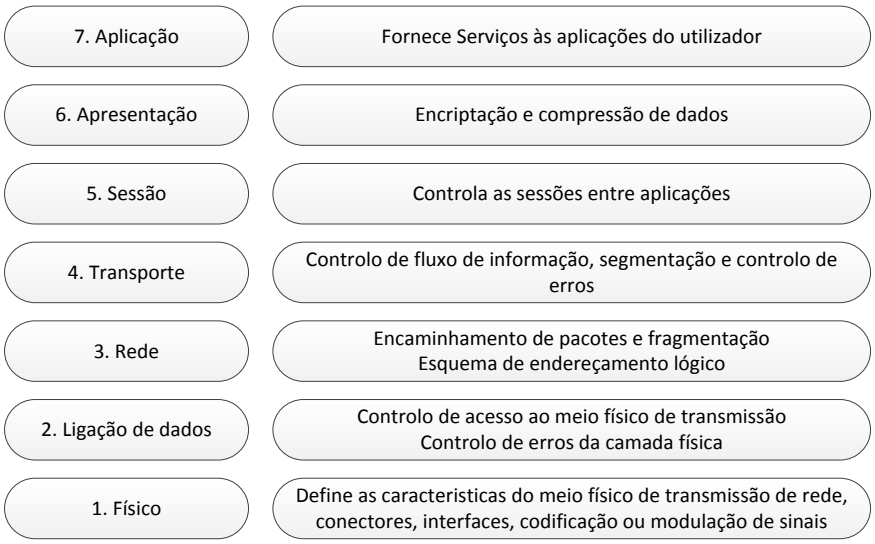

Figura 3.6 Camadas do modelo OSI

3.4.1 Camadas

É na camada de transporte que se processa a transferência de dados entre dispositivos, independentemente da topologia, tipo ou configuração das redes físicas existentes entre eles. Os protocolos mais relevantes desta camada são exatamente o protocolo TCP e UDP.

3.4.2 UDP/TCP

O protocolo UDP surgiu como uma alternativa ao TCP, quando este apenas permitia circuitos virtuais. Numa versão posterior, o TCP foi reorganizado, sendo bipartido no *Internet Protocol* (IP), responsável pelo endereçamento e roteamento de pacotes individuais e no TCP, responsável pelo controlo do fluxo de dados e pela recuperação de pacotes perdidos, dando assim origem ao protocolo conhecido, TCP/IP. O protocolo UDP, no endereçamento de pacotes de dados proveniente da camada rede, revela-se como um complemento ao protocolo IP.

Enquanto o protocolo IP é responsável pelo endereçamento lógico, permitindo também a transmissão de pacotes entre duas máquinas conectadas na mesma rede, o protocolo UDP providencia um endereçamento suplementar que permite a criação de várias ligações para diversas aplicações presentes numa máquina, sendo-lhes definido um ponto de acesso (porto) distinto para cada tipo de aplicação.

O protocolo TCP difere, no entanto, do protocolo UDP, pelo facto de este ser um protocolo orientado para a ligação, abrangendo diversos mecanismos de verificação de pacote recebido. Por seu turno, o protocolo UDP não assume qualquer controlo e necessidade de gerar uma ligação entre cliente e servidor.

As suas principais vantagens devem-se a ser uma escolha adequada para transferência de dados em tempo real, suporte de *broadcasting* e *multicasting*, não desperdiçando tempo na criação ou destruição de conexões realizadas.

3.4.2.1 Pacote UDP

Na Figura 3.7 encontra-se representado o formato de um pacote de dados UDP.

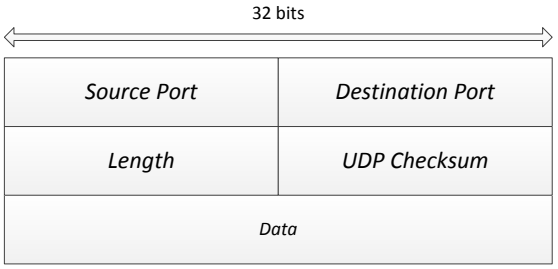

Figura 3.7 Descrição de pacote de dados UDP

Cada elemento tem um tamanho de 16 bits, excetuando o campo referente aos dados a enviar. Tanto o valor de *Source Port* como o de *Checksum* são campos opcionais no caso de IPv4, o que não se verifica no IPv6 onde apenas o *Source Port* representa um campo facultativo.

O campo ***Source Port*** destina-se ao porto de saída pelo qual os dados são enviados.

Ao valor referente ao campo ***Destination Port*** deverá corresponder o porto de entrada, por onde os dados enviados deverão ser recebidos.

No campo ***Length*** deverá ser introduzido a informação acerca do número total de bytes do pacote a ser enviado. O valor poderá variar entre 0 e 65535 (2^{16}), no entanto o seu valor mínimo considerado será 8, devido a este corresponder à dimensão do *header*.

O campo ***Checksum*** é utilizado como teste de erros na transmissão de pacotes. Para realizar o teste é adicionado um "*pseudoheader"* com os dados referentes ao protocolo IP. Após o reagrupamento dos blocos, estes são organizados em blocos de 16 bits. O valor respetivo do *checksum* resultará da soma de todos os blocos, realizando posteriormente complemento para um do valor resultante, sendo este armazenado no *checksum*. Quando o pacote chega ao seu destino, o procedimento é repetido. Somando o valor atual ao anterior e, no caso de o valor ser igual a 0, considera-se a não existência de erros de transmissão.

3.4.3 Telnet

O protocolo *Telnet* [38] é um protocolo de rede presente na camada da aplicação (camada 5). Frequentemente utilizado para facilitar a comunicação de controlos entre dois terminais, funciona como interface de dois terminais interligados por uma rede de internet. Baseando-se numa conexão TCP para encaminhar os dados em formato ASCII codificados em 8 bits, fornece assim um sistema orientado para comunicação bidirecional (*half-duplex*).

3.4.4 FTP

Outros protocolos inseridos numa conexão de TCP/IP (ex., FTP – *File Transfer Protocol*) baseiam-se no protocolo *Telnet*. Não existindo uma autenticação de ambos os terminais, é considerado um protocolo de transferência de dados não seguro. A transmissão de dados consiste unicamente em transferir os bytes no fluxo de TCP, sendo previamente armazenados num *buffer*.

3.4.5 UART

UART é um canal de comunicação série assíncrona *full-duplex* permitindo que se interliguem dois terminais. Para tal, é frequente a utilização de *standards* de comunicação tais como *RS-232*.

A UART é essencialmente constituída por 3 módulos distintos: Um módulo de transmissão (TX), um modo de receção (RX) e um gerador do sinal de relógio para transmissão e receção, habitualmente designado de *baud rate*.

A comunicação UART permite assim:

- Conversão série-paralelo de dados enviados por um periférico
- Conversão paralelo-série de dados transmitidos por um periférico

O processador recebe e escreve os dados sendo estes armazenados em *buffers* (FIFOs) de entrada ou de saída, aptos para enviar/receber dados até 32 bits.

Na transmissão, os dados são transmitidos a partir da FIFO. Caso a UART seja ativada, é gerada uma *frame* de início de transmissão. É ainda definido o tempo de relógio (*baudrate*), que determina a taxa de transmissão das mensagens.

Na receção, os sucessivos caracteres recebidos na linha série são armazenados no *buffer* de receção, processo semelhante ao de transmissão.

Na Figura 3.8, é possível observar a *frame* de dados de um pacote de a enviar pela UART.

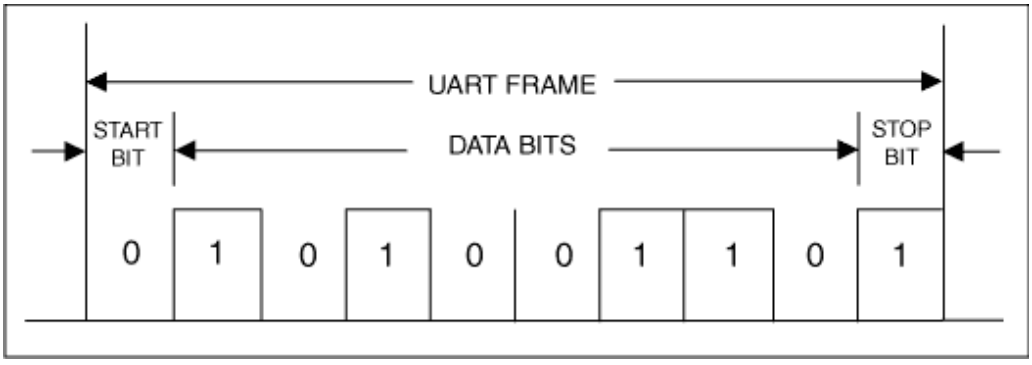

Figura 3.8 *Frame* de dados de um bloco de dados pertencente à UART [39]

3.5 API

O *AR.Drone 2.0* é um *quadcopter* sendo portanto inerentemente instável, depositando fortemente no processador *on-board* a capacidade para realizar a estabilização e para fornecer a assistência necessária ao utilizador no processo de movimento. Para além da assistência providenciada, é também garantida a comunicação necessária para que este seja controlado externamente por um dispositivo remoto.

Apesar de ser possível realizar algumas comunicações básicas via *telnet*, o *firmware* de controlo não está disponível em *open-source* nem documentado, o que faria com que uma possível tentativa de alterar o código-fonte interno pudesse danificar permanentemente o *drone*. Como tal, é disponibilizada uma API onde são fornecidos os comandos necessários para configuração e controlo de navegação, encarando o *drone* como uma caixa negra, utilizando apenas os canais e interfaces de comunicações disponíveis para aceder às suas funcionalidades.

O *AR.Drone* sendo capaz de efetuar uma comunicação *Wi-Fi* b,g,n possui a capacidade de alcance máximo de cinquenta metros e realiza *uplink* e *downlink* a cerca de 100Mbit/s.

Um dispositivo pode ser conectado ao *drone* utilizando uma rede *Wi-Fi* e atribuindo-lhe um endereço IP a partir do servidor DHCP do *drone*.

No momento em que a ligação é gerada, o controlador externo é capaz de comunicar com o *drone* utilizando quatro canais distintos, com quatro portos UDP distintos:

- *Navigation channel* (porto UDP 5554);
- *Video channel* (porto UDP 5555);
- *Command channel* (porto UDP 5556);
- *Control port* (porto TCP 5559, opcional).

3.5.1 Canal de Transmissão de Dados de Navegação

Operando no modo normal, o *drone* apenas devolve dados de navegação básicos a cada 30 milissegundos. No momento em que é comutado para o modo de *debug*, este inicia a transferência de uma grande quantidade de informação proveniente dos sensores a cada 5 milissegundos. Não sendo uma prioridade a forma como esses dados são codificados (estando essa informação documentada [25]), os parâmetros essenciais das leituras dos sensores a ter em conta são:

- **Orientação** (os valores de *roll*, *pitch*, *yaw*, bem como os seus respetivos ângulos): os valores dos ângulos de *roll* e *pitch* são bastante precisos e pouco variáveis, o que não se verifica para o valor de *yaw*, estando sujeito a variações ao longo do tempo.
- **Velocidade horizontal**: Com o propósito de possibilitar ao *drone* manter a sua posição apesar do possível vento, é gerado internamente um fluxo ótico baseado num algoritmo de decisão através da câmara vertical para determinar a velocidade horizontal do *drone*. No entanto, sabe-se que a precisão dos valores medidos depende fortemente do piso que sobrevoa: se este é texturizado ou não; quando sobrevoa um piso texturizado (como por exemplo uma secretária desarrumada) os valores são altamente precisos, no entanto, quando este se encontra a sobrevoar uma superfície com pouca textura, a qualidade da velocidade estimada é claramente errónea, desviando-se do real valor de cerca de 1 m/s.
- **Altura** (medida em milímetros): Este valor é baseado exclusivamente nas leituras do sensor de ultrassons. Enquanto o *drone* se encontrar a sobrevoar superfícies planas e reflexíveis, este valor apresenta uma precisão com um erro de 8,5 cm por cada metro de altura. No entanto, quando se encontra a sobrevoar superfícies irregulares ou próximo de paredes, o valor da altitude pode estar subjugado a grandes flutuações.

  Esta situação geralmente provoca alterações súbitas na aceleração vertical, altamente indesejada, com o *drone* a tentar manter a sua altura relativa julgando tratar-se da sua altura absoluta. Este valor é medido a cada 40 milissegundos.
- **Estado da bateria**: um valor inteiro entre 100 e 0, representando a percentagem de bateria disponível.
- **Estado de controlo** (campo de 32 bits): indica o estado de voo. Poderá ter como valores, a título ilustrativo, de "*LANDED*", "*HOVERING*", "*ERROR*", "*TAKE OFF*", etc.
- **Tempo de Atualização:** valor interno referente ao tempo a que os dados foram transmitidos, em microssegundos. Este não é necessariamente o tempo em que as leituras dos sensores foram efetuadas, sendo que num mesmo pacote, poderão coexistir medições com diferença temporal de cerca de 60 milissegundos.

3.5.2 Canal de Transmissão de Vídeo

Através do canal de vídeo é possível transmitir imagens da câmara frontal e da câmara vertical. A imagem da câmara frontal não é concebida na resolução máxima da câmara, estando numa escala inferior e comprimida para reduzir o seu tamanho e para acelerar o processo de transferência via *Wi-Fi*.

Como tal, o controlador externo obtém uma imagem de 320x240 pixéis com uma profundidade de cor de 16 bits. Uma pequena desvantagem do sistema de câmaras deve-se ao facto de não ser possível obter a imagem de ambas as câmaras em simultâneo. Devido a este facto, o utilizador deverá escolher entre a câmara frontal e a câmara vertical. A mudança não é instantânea podendo demorar cerca de 300 milissegundos e durante a sua transferência, a imagem fornecida poderá conter dados inválidos.

3.5.3 Canal de Transmissão de Comandos

O *AR.Drone* é controlado através do *streamig* de pacotes contendo comandos, em que cada um define os seguintes parâmetros:

1. Os ângulos de *roll* e *pitch* desejados, a velocidade rotacional de *yaw*, tal como a velocidade vertical *gaz*, em que cada um representa a fração de um máximo permitido, i.e. de um valor entre -1 e 1;
2. Um bit alternando entre o *hover-mode* (modo em que o *drone* tenta manter a sua posição, ignorando qualquer outro comando de controlo) e o modo de controlo manual;
3. Um bit indicando a existência (ou não) de um *error state*, desligando seguidamente todos os motores (*emergency mode*), em caso de estar ativado;
4. Um bit ordenando a descolagem ou aterragem.

Os comandos de controlo são enviados através de um porto UDP, ou seja, a receção de qualquer pacote que contenha um comando de controlo não poderá ser garantido. Desta forma é implementada uma função em que o comando é reenviado aproximadamente a cada 10 milissegundos, permitindo um controlo suave.

3.5.4 Porto de controlo

Sendo o microprocessador interno acessível através de *telnet*, o utilizador poderá conectar-se ao *drone* e alterar as configurações do sistema operativo *on-board*.

Além do mais, é ainda possível realizar um *cross-compil*e de uma aplicação para o processador ARM e corrê-la diretamente na *main board* do *AR.Drone*. Neste caso, é possível aceder às câmaras do *drone* e aos sensores internos sem o *delay* provocado pela transferência de dados via *Wi-Fi*.

Desta forma, é possível alcançar ciclos de controlo mais curtos de execução a um nível baixo de controlo.

Mesmo quando uma aplicação personalizada está a correr na plataforma da *motherboard*, os controlos internos responsáveis pela estabilidade, podem ser ativados. Contudo, é necessário que a memória e os limites computacionais da *motherboard* sejam tidos em conta quando se está a desenvolver uma aplicação com o objetivo de correr internamente no microprocessador.

Sob pena de não danificar o *drone* internamente, foi concebida uma aplicação que externamente armazena as leituras efetuadas pelos sensores e processos de transmissão de comandos, reduzindo, assim, o risco de danificar fisicamente o *drone*, correspondendo a uma solução mais amigável e de maior simplicidade de compreensão.

Este tipo de aplicação poderá também servir de base para aplicações que venham a ser desenvolvidas de uma forma mais complexa. O *software* desenvolvido é completamente independente das bibliotecas internas do *drone* e funciona num ambiente *GNU/Linux*.

3.6 Controlador de Navegação

A comunicação com o *drone* é essencial no processo de navegação autónoma, sendo necessário o envio de comandos que sejam validados pelo seu *firmware* e posteriormente executados.

Não estando o sistema desenvolvido sob o *Software Development Kit* (SDK) fornecido pela empresa produtora, é necessário que as rotinas de configuração e controlo sejam substituídas por comandos AT.

3.6.1 Comandos AT

Os comandos AT são *strings* de texto remetidas para *drone*, com a propósito de controlar as suas ações. Estes poderão ser enviados diretamente em pacotes UDP através do porto 5556. Um controlo tolerável é conseguido enviando comandos AT a cada 30 milissegundos, por forma a obter movimentos suaves. Para precaver possíveis perdas de dados nas transmissões realizadas, dois comandos consecutivos deverão ser enviados em menos de 2 segundos.

3.6.1.1 Sintaxe

As *strings* são codificadas como caracteres ASCII de 8 bits, concluídas com um caráter de *carriage return* (valor de bit $0D_{(16)}$) representado daqui em diante como marcador de nova linha <CR>.

Um comando inicia-se com três caracteres: AT* (i.e., três *words* de 8 bits com os valores $41_{(16)}$, $54_{(16)}$ e $2A_{(16)}$), seguidos do tipo de comando, o número de sequência e opcionalmente, uma lista de argumentos separados por vírgulas dependendo do comando de controlo desejado.

Um pacote UDP poderá conter um ou mais comandos, separados pelos marcadores de nova linha <CR>. Um comando AT deverá transmitir-se apenas num pacote UDP, não sendo possível separá-lo em dois ou mais pacotes.

Exemplo:

*AT*PCMD_MAG = 21625,1,0,0,0,0,0,0<CR>AT*REF = 21625,290717696<CR>*

O tamanho máximo de um comando não poderá exceder os 1024 caracteres, caso contrário será inteiramente rejeitado. Comandos AT com sintaxe errada não serão considerados, não obstante ser confirmado pelo utilizador se os comandos são enviados corretamente através de um analisador de pacotes UDP.

Os argumentos poderão ser do tipo:

- Inteiro com sinal, armazenado numa *string* com representação decimal (ex: o número de sequência);
- Caráter armazenado numa *string* entre " " (ex.: os argumentos do comando *AT*CONFIG*);
- Vírgula flutuante (*float*). Estes não poderão ser armazenados diretamente num comando de *string*. Como alternativa, uma *word* de 32 bits contendo o valor em *float*, deverá ser considerada como um inteiro de 32 bits com sinal e armazenado no comando AT.

3.6.1.2 Sequência de Comandos

Para evitar a execução de comandos já processados, um número sequencial é associado a cada comando AT, sendo o primeiro valor armazenado após a introdução do caráter "=". O *drone* não executará qualquer outro comando com um número de sequência menor do que o valor de sequência do último comando válido recebido. Este valor será reiniciado e igualado a "1" sempre que o piloto se desconectar do porto UDP referente ao envio de comandos AT (a desconexão é realizada após não ter sido enviado qualquer comando num período de 2 segundos) ou quando um comando é enviado com o valor da sequência fixado em 1. O sistema deverá respeitar estas regras sob pena de executar os comandos com sucesso:

- Enviar sempre o valor da sequência a 1 no primeiro comando enviado;
- Incrementar continuamente o valor da sequência a cada comando enviado, após o primeiro.

3.6.1.3 Parâmetros em Vírgula Flutuante

Considerando um argumento em vírgula flutuante para o valor de *pitch*, de -0.8. O valor será armazenado em memória como uma *word* de 32 bits cujo valor é $BF4CCCCD_{(16)}$, de acordo com o formato IEEE-754 [40]. A *word* deverá armazenar o valor inteiro de 32 bits, $-1085485875_{(10)}$.

Logo o comando deverá ser *AT*PCMD_MAG=xx,xx,-1085485875,xx,xx,xx,xx*.

3.6.1.4 Comandos

Existem vários tipos de comandos, no entanto os mais utilizados estão representados na Tabela 3.1.

Tabela 3.1 Resumo dos comandos AT [25]

Comando AT	Argumentos	Descrição
*AT*REF*	*input*	*Take Off/Land/Emergency stop command*
*AT*PCMD*	*flag, roll, pitch, gaz, yaw*	*Move the drone*
*AT*PCMD_MAG*	*flag, roll, pitch, gaz, yaw, psi, psi accuracy*	*Move the drone (with Absolute control support)*
*AT*FTRIM*	-	*Set the reference for the horizontal plane (must be on ground)*
*AT*CONFIG*	*key, value*	*Configuration of the AR.Drone 2.0*
*AT*CONFIG_IDS*	*session, user, application ids*	*Identifiers for AT*CONFIG commands*
*AT*COMWDG*	-	*Reset the communication watchdog*
*AT*CALIB*	*device number*	*Ask the drone to calibrate the magnetometer (must be flying)*

A descrição detalhada dos comandos AT existentes é efetuada no anexo A.

3.6.2 Dados de Navegação

Os dados de navegação (*navigation data* ou *navdata*) representam a informação que a aplicação irá receber periodicamente (menos de 5 ms) sobre os estados do *drone* (ângulos, altitude, câmara, velocidade, deteção de *tags*, etc..).

Os dados de navegação serão enviados através do porto *UDP* 5554. A informação é armazenada num formato binário descrito na Tabela 3.2 consistindo em vários blocos denominados de opções.

Cada opção contém um *header* (2 bytes) identificando o tipo de informação armazenada, um inteiro de 16 bits que identifica o tamanho dos blocos, e vários tipos de informação coletada como inteiros de 32 bits, números em vírgula flutuante de 32 bits ou *arrays*. Toda a informação recebida é armazenada em formato *little-endian* (*byte* menos significativo armazenado no menor endereço).

Tabela 3.2 Composição de um pacote de *navdata* [25]

Header 0x55667788	***Drone State***	***Sequence number***	***Vision flag***		***Option 1***				**…**		***Checksum block***		
					Id	***Size***	***Data***		**…**		***Cks id***	***Size***	***Cks data***
32 bit Int	*32 bit Int*	*32 bit Int*	*32 bit Int*		*16 bit Int*	*16 bit Int*	*… …*		*… …*		*16 bit Int*	*16 bit Int*	*32 bit int*

3.6.2.1 Receção de Dados de Navegação

Para receber devidamente os dados de navegação, é necessário que sejam enviadas as configurações iniciais para o porto referente, descritas na Figura 3.9. Existem dois estados possíveis:

- O *drone* inicia em modo *bootstrap* (modo autossustentável, não recebendo qualquer comando exterior), sendo apenas atualizado o seu estado e o número de sequência;
- O *drone* está já iniciado, sendo enviados os estados essenciais de navegação.

Para desbloquear o modo de *bootstrap*, permitindo que a transmissão de comandos e receção de dados se processe de forma normal, é executada a sequência de comandos presente na Figura 3.9.

Inicialmente deverá ser enviado um pacote de dados para o *drone* para que este termine o processo de *bootstrap* e inicie o processo de transmissão de *navdata*, enviando de volta uma resposta.

De seguida é enviado um comando ativando o envio de dados essenciais de navegação para o *drone*. Este responderá em caso de sucesso, através do porto, a *flag* "*ARDRONE_COMMAND_MASK"* ativa.

Finalmente, é enviado um comando AT com o bit "*ACK_CONTROL_MODE"* ativo, obtendo como resposta o envio de *navdata* continuamente.

3.6.2.2 Sincronização

A sincronização da transmissão de dados é fundamental, evitando que mensagens erradas ou comandos não identificados sejam recebidos.

Existem dois casos particulares em que o controlador poderá fazer um *reset* ao número de sequência (e consequentemente um *reset* ao sistema):

- O *drone* não reportar qualquer informação durante um período de 50 ms: o bit de "*ARDRONE_COM_WATCHDOG_MASK*" do pacote de *navdata* é então ativado. Para sair deste modo, o utilizador deverá enviar um comando *AT*COMWDG*, efetuando um *reset* ao *timer watchdog*;
- O *drone* não receber qualquer comando de controlo durante um período de 2000 ms: A comunicação é posteriormente desligada, sendo que o bit *ARDRONE_COM_LOST_MASK* será atualizado para "1". O utilizador deverá então, reiniciar a ligação.

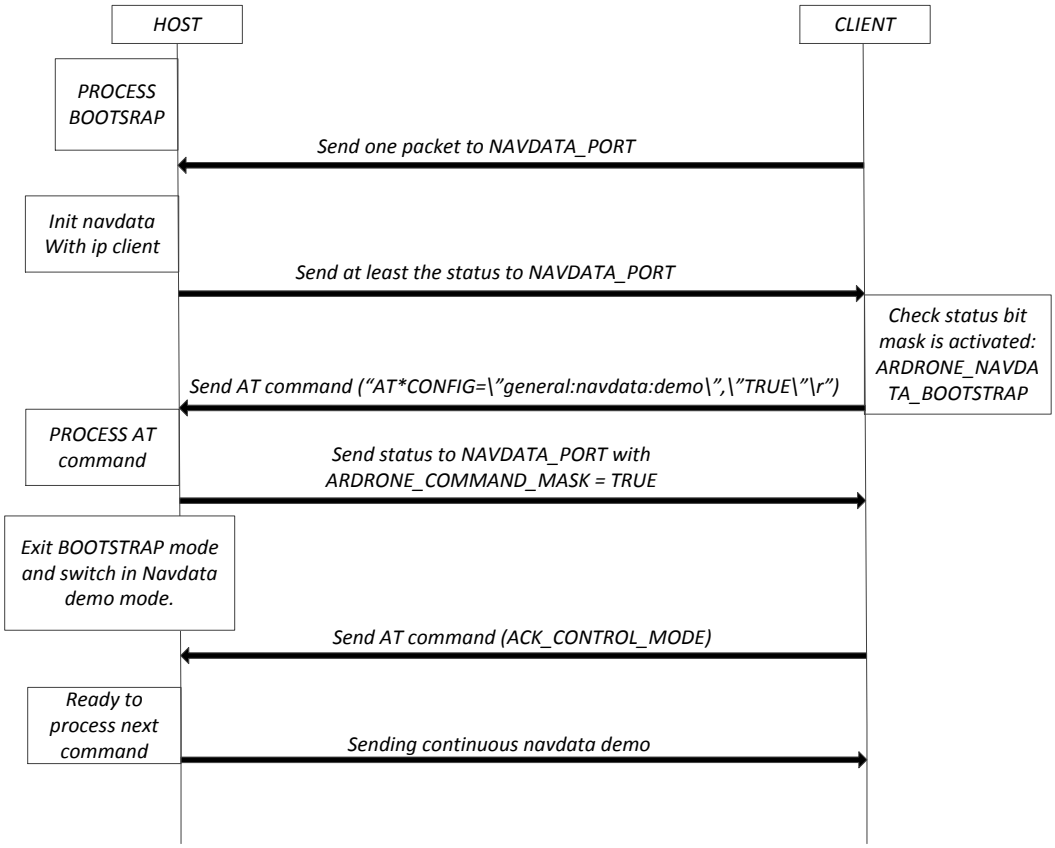

Figura 3.9 Inicialização do processo de transmissão de dados de navegação [25]

3.6.3 Configuração Inicial

Para o correto funcionamento do *drone*, é necessário que seja realizada uma configuração inicial de vários parâmetros, através do comando *AT*CONFIG*.

O comando deverá ser enviado com o número de sequência correspondente e o respetivo parâmetro entre " ", seguido do seu valor (também ele entre " "). Estas configurações deverão ser enviadas antes de o *drone* descolar.

A descrição dos comandos é detalhada no anexo A.

Capítulo 4

Plataforma de Navegação

Neste capítulo são abordados os conceitos essenciais na plataforma de comunicação.

Será realizada uma descrição detalhada das plataformas físicas, bem como das comunicações realizadas entre estas.

Finalizar-se-á com uma descrição do modo de leitura e processamento de mensagens provenientes do recetor GPS.

4.1 Recetor GPS

Um recetor de GPS é um módulo de receção de mensagens GPS, com a capacidade de receber vários formatos de mensagens. É por isso indispensável compreender a sua forma de operação.

4.1.1 Formato Padrão de Mensagens GPS

A tarefa de combinar os dados de vários recetores tornou-se bastante árdua, desde que os primeiros fabricantes individuais de recetores GPS criaram os seus próprios formatos de mensagens para armazenamento dos sinais.

Foi então que, perante tais obstáculos, um grupo de investigadores decidiu desenvolver padrões de mensagens para utilização em recetores GPS. Os padrões mais utilizados são nomeadamente RINEX, NGS-SP3, RTCM SC-104 e NMEA 0183. Detalhar-se-á o formato NMEA 0183, pois será aquele que irá dar suporte ao sistema, como formato de mensagem GPS a ser recebida.

4.1.1.1 Formato de Mensagem NMEA 0183

A *Nation Marine Electronics Association* (NMEA) foi fundada em 1957 por um grupo de comerciantes de material eletrónico, com o objetivo de fortalecer as suas relações com os fabricantes de componentes eletrónicos [14].

Em 1983, com a admissão dos fabricantes e das organizações privadas e governamentais, foi adotado como formato modelo de interligação entre dispositivos eletrónicos.

Os padrões do formato NMEA 0183 [41] são *streams* de dados em codificação ASCII, transmitidos a um ritmo variável, através de um dispositivo capaz de remeter as mensagens para outros dispositivos (ex., de um recetor GPS para um computador).

O *stream* de dados terá o formato de mensagem em que cada nova mensagem será iniciada com o caráter "$" e concluída com o terminador de transmissão, <CR><LF>. O caráter de nova mensagem "$" é seguido de cinco caracteres indentificadores do tipo de mensagem a receber (ex., *GPGGA*) seguidos do tipo de dados e das definições específicas da mensagem. O último parâmetro em qualquer mensagem recebida é o *checksum*, após a introdução do delimitador de dados, "*". O número máximo de caracteres em qualquer mensagem é de 82 caracteres, o que corresponde a 79 caracteres entre o caráter de início e o terminador.

Existe uma panóplia de mensagens distintas, no entanto são selecionados apenas 5 tipos de mensagens descodificadas pelo *hardware*, descritas na Tabela 4.1.

Tabela 4.1 Tipos de mensagens NMEA [41]

GPGGA	*GPS Fix Data* – Mensagem que contém dados referentes ao tempo e posição do recetor GPS na ligação com o satélite.
GPGSA	*GPS DOP and Active Satellites* – Mensagem referente ao modo de operação, satélites ativos na posição atual e valores de precisão de posição (valores de DOP).
GPGSV	*GNSS Satellites in view* – Mensagem referente ao número de satélites disponíveis do ponto de vista dos valores de ID, elevação, *azimuth*, e SNR.
GPRMC	*Recommended Minimum Specific GNSS Data* – Mensagem que contém dados referentes ao tempo, data, posição, rota e velocidade. Contém os dados mínimos de navegação.
GPVTG	*Course over Ground and Ground Speed* – Mensagem referente à rota atual e à velocidade relativamente ao solo.

Do conjunto selecionado de mensagens de receção, serão, unicamente, analisados e detalhados os formatos de mensagem GPGGA e GPRMC, por serem os que contém a informação mais adequada e necessária ao sistema de navegação.

GPGGA

A mensagem tem o seguinte formato:

*$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh<CR><LF>*

Exemplo:

*$GPGGA,130305.0,4717.11,N,00833.912,E,1,08,0.94,00499,M,047,M,,*58<CR><LF>*

Na Tabela 4.2 são descritos individualmente todos os parâmetros referenciados na mensagem GPGGA.

Tabela 4.2 Descrição dos termos individualmente do tipo de mensagem GPGGA [41]

$	Início de mensagem
GP	Identidade do Transmissor (GPS)
GGA	Identificador do tipo de dados (*GPS fix data*, neste caso em particular)
,	Delimitador do campo de dados
130305.0	Tempo da posição no sistema UTC: 13h 03min 05 seg
4717.115	Latitude: 47º 17.115 min
N	Latitude direcionada para Norte (N = Norte, S = Sul)
00833.912	Longitude: 8º 33.912 min
E	Longitude orientada para Este (E = Este, W = Oeste)
1	Indicador de qualidade do GPS: 0 – Sem sinal GPS 1 – GPS com *C/A-code* 2 – DGPS com *C/A-code*
08	Número de satélites fixado no envio da mensagem
0.94	*Horizontal Dilution of Precision* (HDOP)
00499	Altura da antena em relação ao nível do mar
M	Metros (unidades em altura)
047	Diferença de altura entre uma elipsoide e um geoide
M	Metros (unidades de separação geoide)
”	Tempo do DGPS em segundos (tempo desde que a última mensagem do tipo RTCM entre 1 e 9 foi recebida; Campo nulo quando o modo DPGS não é usado)
0000	ID da estação referência (no caso do DPGS, usar uma gama entre 0000 e 1,023)
*	Caráter limitador do inico de *checksum*
58	*checksum* (o último *bit* da mensagem)
<CR><LF>	Terminador

GPRMC

As mensagens na configuração RMC têm o seguinte formato:

*$GPRMC,hhmmss.ss,A,llll,ll,a,yyyyy.yy,a,x.x,x.x,xxxxxx,x.x,a,a*hh<CR><LF>*

Exemplo:

*$GPRMC,130304.0,A,4717.115,N,00833.912,E,000.04,205.5,200601,01.3,W,A*7C<CR><LF>*

Na Tabela 4.3 são descritos individualmente todos os parâmetros referenciados na mensagem GPRMC.

Tabela 4.3 Descrição dos termos individualmente do tipo de mensagem GPRMC [41]

$	Delimitador de início de mensagem
GP	Identidade do Transmissor (GPS)
RMC	Identificador de dados (*Recommended Minimum Specific data*, neste caso)
,	Delimitador do campo de dados
130304.0	Tempo da posição no sistema UTC: 13h 03 min 04.0 seg
A	*Status*: A – Válido V – Inválido
4717.115	Latitude: 47º 17.115 min
N	Latitude direcionada para Norte (N = Norte, S = Sul)
00833.912	Longitude: 8º 33.912 min
E	Longitude orientada para Este (E = Este, W = Oeste)
000.04	Velocidade horizontal: 0.04 nós
205.5	Direção: 205.5º (Relativamente ao Norte(0º))
200601	Data: 20 de Junho 2001
01.3	Ajuste de inclinação: 1.3º
W	Declive da direção ocidental (E = Este, W = Oeste)
*	Caráter limitador do inico de *checksum*
A	Indicador do Sistema de posicionamento: A – modo autónomo D – modo diferencial E – modo estimado
7C	*checksum* (o último *bit* da mensagem)
<CR><LF>	Terminador

4.1.2 Módulo Físico do Recetor GPS

O recetor GPS utilizado no projeto foi o MEDIATEK – 3329 compactado num *System on Package* (SOP) que consta na Figura 4.1. O recetor é capaz de um elevado nível de precisão na leitura do posicionamento e na velocidade de receção, com uma grande capacidade de sensibilidade e rastreamento em circunstâncias urbanas. O módulo suporta 66 canais e é utilizado para aplicações com necessitem do auxílio de recetores GPS como:

- Gestão de frotas;
- Sistemas de *Location-based service* (LBS) e *Auto Vehicle Location* (AVL);
- Sistemas de segurança;
- Dispositivos para posicionamento pessoal e navegação em viagens.

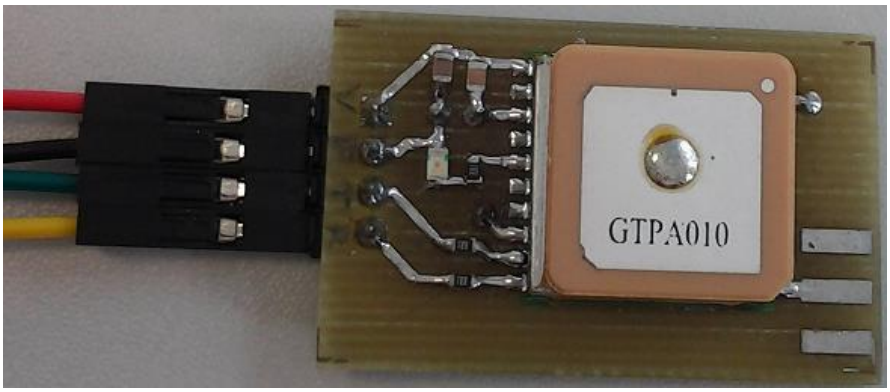

Figura 4.1 SOP com antena do recetor GPS *Mediatek - 3329*

4.1.2.1 Características

- *MediaTek MT3329 Single Ship;*
- Frequência L1, *C/A-code*, 66 canais;
- Suporte acima de 210 de canais PRN;
- Deteção e redução de *Jammer;*
- Deteção e compensação de *Multi-path*;
- Dimensões: 16 mm x 16 mm x 6 mm;
- Dimensões do SOP da antena: 15 mm x 15 mm x 4 mm;
- Sensibilidade acima de *-165 dBm* no *tracking*, com desempenho superior em zonas urbanas, comparado com outros módulos;
- Precisão no posicionamento*:*
    - Sem suporte*:* 3m 2D – RMS;
    - DGPS *(*RTM, SBAS(EGNOS))*:* 2.5m 2D-RMS;
- Baixo consumo de energia*:* 48 mA @ aquisição, 37 mA *@ tracking;*

- Baixo consumo de energia em *Shut-Down:* 15 µA, tipicamente;
- Suporte DGPS (EGNOS) (Ativo por defeito);
- Frequência máxima de atualização: acima de 10Hz (Configurável através do *firmware*);
- Interface de suporte USB;
- Acordo com FCC E911 e suporte de AGPS (modo Offline: EPO válida até 14 dias);
- Em conformidade com RoHS;
- Sinal de saída com 8 *bits* de dados, sem paridade e 1 *stop bit*;
- Vários Baud Rates disponíveis. Por defeito, 9600 bps (4800/9600/38400/57600/115200 bps disponíveis através de configuração);
- Protocolo NMEA.

4.1.2.2 Definição dos Portos

Na Tabela 4.4 são caracterizados todos os pinos de entrada do recetor, sendo posteriormente detalhados no anexo B.

Tabela 4.4 Descrição dos portos do recetor GPS [42]

Pin	Designação	I/O	Descrição
1	VCC	PI	Alimentação de entrada DC principal
2	ENABLE	I	Ligar a "1"ou manter em aberto
3	GND	P	*Ground*
4	VBACKUP	PI	*Backup* de alimentação de entrada
5	3D-FIX	O	Indicador de *3D-fix*
6	DPLUS	I/O	Porto *USB* D+
7	DMINUS	I/O	Porto *USB* D-
8	GND	P	Ground
9	TXD	O	Porto de saída de dados no formato NMEA
10	RXD	I	Porto de entrada de atualizações de *firmware*

4.1.2.3 *Design* de Referência

Na Figura 4.2 observa-se as ligações externas no SOP, tal como o design de referência utilizado para transferência de mensagens GPS através da UART.

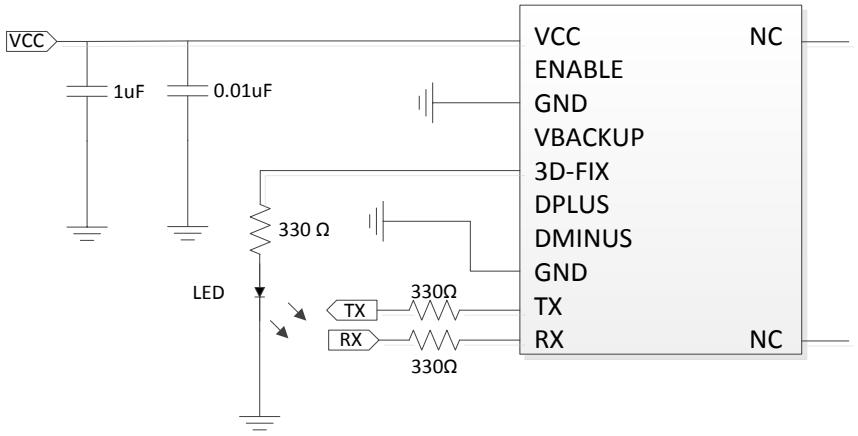

Figura 4.2 *Design* de referência do SOP na configuração UART [42]

4.1.2.4 Características DC

Na Tabela 4.5 é possível observar as características DC do recetor.

Tabela 4.5 Características DC do Recetor [42]

Parâmetro	Condição	Min.	Tipico.	Max.	Unid.
Operation Suply Voltage	-	3.2	3.3	5.0	V
Operation Suply Ripple Voltage	-	-	-	40	mVpp
Backup Battery Voltage	-	2.0	3.0	4.3	V
RXA TTL H Level	VCC = 3.3V	2.1	-	2.8	V
RXA TTL L Level	VCC = 3.3V	0	-	0.9	V
TXA TTL H Level	VCC = 3.3V	2.1	-	2.8	V
TXA TTL L Level	VCC = 3.3V	0	-	0.8	V
USB D+	VCC = 5.0V				V
USB D-	VCC = 5.0V				V
Power Consuption @ 3.3V	*Acquisition*	43	48	53	mA
	Tracking	32	37	42	mA
Backup Power Consumption @ 3.0 V	25ºC	-	10	-	uA
Shut-down Power Consumption (via enable pin)	25ºC	-	15	-	uA

4.2 *Arduino*

Como interface de receção e descodificação das mensagens recebidas provenientes do recetor GPS foi utilizado o microcontrolador *Arduino Nano* [43], cujo *design* é observável na Figura 4.3.

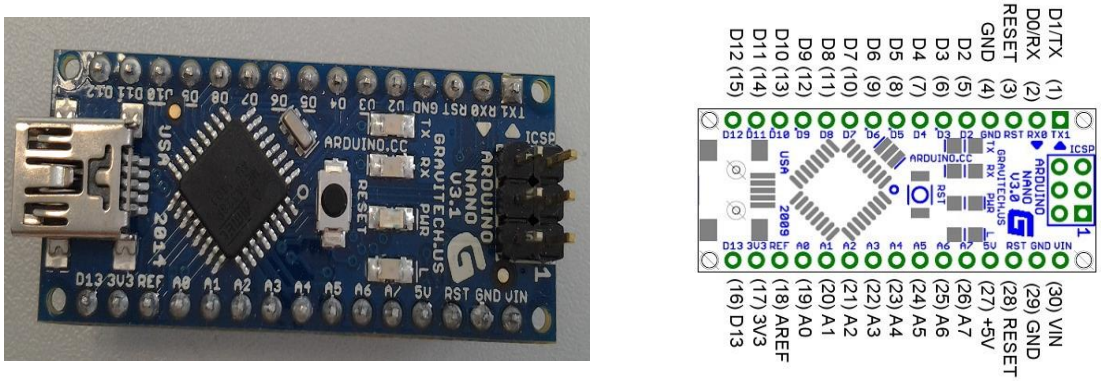

Figura 4.3 *Arduino Nano* e respetivo *pinout* [44]

Devido à sua pequena dimensão e excelente portabilidade, o microcontrolador permite reduzir o peso da carga depositado no *drone*, contrariamente a outros. Devido ao seu baixo consumo de energia, permite também que sejam realizadas várias horas de voo, sem necessidade de trocar a fonte de alimentação (Pilha de 9 V). Processando as necessidades essenciais para controlar autonomamente um *drone* via GPS, apresenta-se como a solução ideal para o desenvolvimento do sistema, com um preço próximo dos 30€.

4.2.1 Especificações Físicas do *Arduino Nano*

4.2.1.1 Especificações

- Microcontrolador: ATmel ATmega328;
- Tensão de Operação (*logic level*): 5 V;
- Tensão de entrada (recomendada): 7-12 V;
- Tensão de entrada (limites): 6-20 V;
- Portos digitais I/O: 14 (em que 6 fornecem uma saída *Pulse Width Modulation* (PWM));
- Portos analógicos I/O: 8;
- Corrente DC por porto I/O: 40 mA;
- Memória *Flash*: 32 KB em que 2 KB são utilizados pelo *bootloader;*
- SRAM: 2 KB;
- EEPROM: 1 KB;
- Velocidade de relógio: 16 MHz;
- Dimensões: 18 mm x 43 mm.

4.2.1.2 Alimentação

O *Arduino Nano* permite ser alimentado através de uma porta *Mini-USB* que por 6-20 V, não regulados e alimentados externamente através do porto 30; quer por 5 V, regulados, também com alimentação externa através do porto 27. A fonte de alimentação é automaticamente comutada para a fonte de tensão mais alta.

O *chip* FTDI FT232RL no *arduino* é apenas ativado na condição de a placa estar a ser alimentada através da porta USB. Como tal, quando alimentada externamente, a saída a 3.3V (que está a ser alimentada pelo chip FTDI) não está disponível e os LEDs RX e TX irão piscar caso os portos digitais 0 ou 1 estejam ligados.

4.2.1.3 I/O

Cada um dos 14 portos digitais pode ser utilizado como entrada ou saída. Operando a 5 Volts, cada porto é capaz de fornecer ou receber um máximo de 40 mA e está equipado internamente com uma resistência *pull-up* de 20-50 KΩ.

Porém, alguns portos contêm funções específicas:

- Portos 0 (RX) e 1 (TX): comunicação série. Utilizados para receber (RX) e transmitir (TX) dados via TTL série. Estes portos estão conectados aos correspondentes portos do chip FTDI USB-*to*-TTL série;
- Portos 2 e 3: Interrupções externas. Estes portos podem ser configurados para forçar uma interrupção num "0", em flancos ascendentes ou descendentes, ou apenas com a alteração de sinal;
- Portos 3, 5, 6, 9, 10 e 11: PWM;
- Porto 13: LED. Quando o porto está a "1", o LED está ligado; quando o porto está a "0", o LED encontra-se desligado;
- Portos 10 (SS), 11 (MOSI), 12 (MISO) e 13 (SCK): SPI. Estes portos suportam comunicação SPI que embora esteja incluído no hardware, não se encontra para já disponível na linguagem *arduino*.

Os oito portos analógicos que individualmente proporcionam 10 *bits* de resolução (1024 valores diferentes), na sua configuração padrão permitem medir entre 0 a 5 Volts, embora seja possível proceder uma alteração dos valores padrão. Os portos analógicos 6 e 7 não poderão ser utilizados como portos digitais. Adicionalmente, os portos têm funcionalidades específicas como:

- Portos 4 (SDA) e 5 (SCL): I2C.

Existem ainda outros portos na placa, nomeadamente:

- AREF: Tensão de referência para as entradas analógicas;
- *Reset*: Tipicamente utilizada para adicionar um botão de *reset* aos *shields* que impossibilitem o acesso à parte física da placa.

4.2.2 Comunicação

O *Arduino Nano* pode comunicar quer com um computador, quer outro *arduino*, quer com outros microntroladores. O microprocessador ATmega328 permite uma comunicação série (UART TTL 5V), disponível nos portos 0 (RX) e 1 (TX). O chip FTDI FT232RL orienta a comunicação série para USB e os drivers FTDI fornecem um porto de comunicação virtual (COM) para o computador. A API do *arduino* inclui um controlador série que permite que dados em formato de texto simples sejam enviados para o microntrolador. Os LEDs RX e TX irão piscar quando os dados estiverem a ser transmitidos entre o chip FTDI e a conexão USB ao computador.

4.2.3 Bibliotecas

O ambiente de programação do *arduino* pode ser ampliado através da inclusão de bibliotecas. Estas facultam funcionalidades para auxiliar no desenvolvimento do código, como por exemplo, no tratamento de dados. O IDE tem já incluída uma gama de bibliotecas padrão, no entanto, outras poderão ser adicionadas ou até criadas.

A biblioteca principal a que o sistema recorre é a biblioteca "*SoftwareSerial*", através da qual, será possível realizar a comunicação série com qualquer microcontrolador.

A biblioteca *TinyGPS* [45] foi utilizada como biblioteca externa do *arduino* e permite efetuar a descodificação de mensagens NMEA transmitidas pelo recetor GPS. Através da biblioteca é possível:

- Processar os caracteres provenientes do GPS;
- Descodificar a latitude e a longitude da posição recebida, bem como a hora de transmissão;
- Devolver a data atual em "*ddmmyy*" e "*hhmmss*";
- Devolver a altura da posição (mensagem GPGCA);
- Determinar a rota da posição (mensagem GPRMC);
- Determinar a velocidade da posição (mensagem GPRMC);
- Calcular o número de satélites captados (mensagem GPRMC);
- Calcular o HDOP da posição;
- Determinar a distância entre 2 posições diferentes;

- Determinar a diferença da rota em graus entre 2 pontos distintos.

4.3 Sistema de Navegação

Como o nome indica, o sistema de navegação autónomo é um sistema que é suportado por sensores combinados com um recetor GPS. Consiste num suporte de *hardware* e *software*, sendo o seu principal objetivo, guiar os *drones* por trajetórias previamente introduzidas, ou navegar através de WP, sendo que um sistema de navegação autónomo funcional deverá ter a capacidade de controlo do *drone* em todas as suas etapas:

- Controlo de altitude;
- Controlo de velocidade;
- Descolagem e aterragem automática;
- Controlo de ângulo de *pitch*;
- Controlo de ângulo de *roll*;
- Coordenação de rotação;
- Controlo de direção.

Para tal, o sistema de navegação será composto por um controlo em *loop*, com o compromisso bipartido de analisar o valor devolvido pelos sensores internos e as mensagens processadas pelo recetor GPS. Posteriormente é efetuada a combinação entre os valores recebidos, traduzindo-se para valores de controlo do movimento, já anteriormente abordados. A Figura 4.4 representa, na sua forma mais simplificada, o ciclo de controlo autónomo do *drone*.

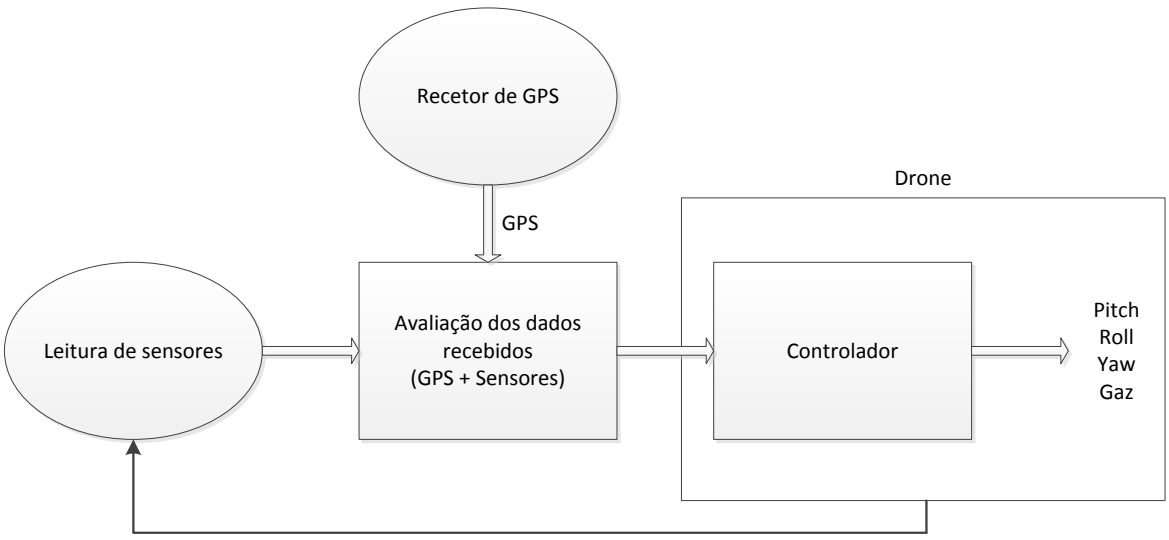

Figura 4.4 Diagrama Básico do sistema de navegação autónomo

O algoritmo de navegação é previamente compilado e armazenado no microcontrolador *arduino* (Avaliador dos dados recebidos). São também pré-carregados todos os WP, com as respetivas latitudes e longitudes definidas. As mensagens GPS são recebidas e descodificadas, sendo transformadas em coordenadas geográficas no formato de graus decimais, para o controlador de voo. O algoritmo irá então calcular a distância e o ângulo da diferença de direção entre o "*home*" e o *waypoint* definido (WP1).

Os respetivos comandos de movimento e controlo, bem como os respetivos parâmetros, são processados em função da distância e da diferença de ângulo relativamente ao destino.

É efetuada uma rotina de receção de mensagens NMEA por forma a evitar que as mensagens transmitidas estejam corrompidas ou incompletas. Como tal, este procedimento (realizado com o auxílio da biblioteca *TinyGPS*) irá analisar a mensagem recebida pelo recetor GPS, certificando-se que o procedimento é efetuado sem erros e com a informação validada. Na Figura 4.5 é detalhado todo o processo de receção de mensagens NMEA.

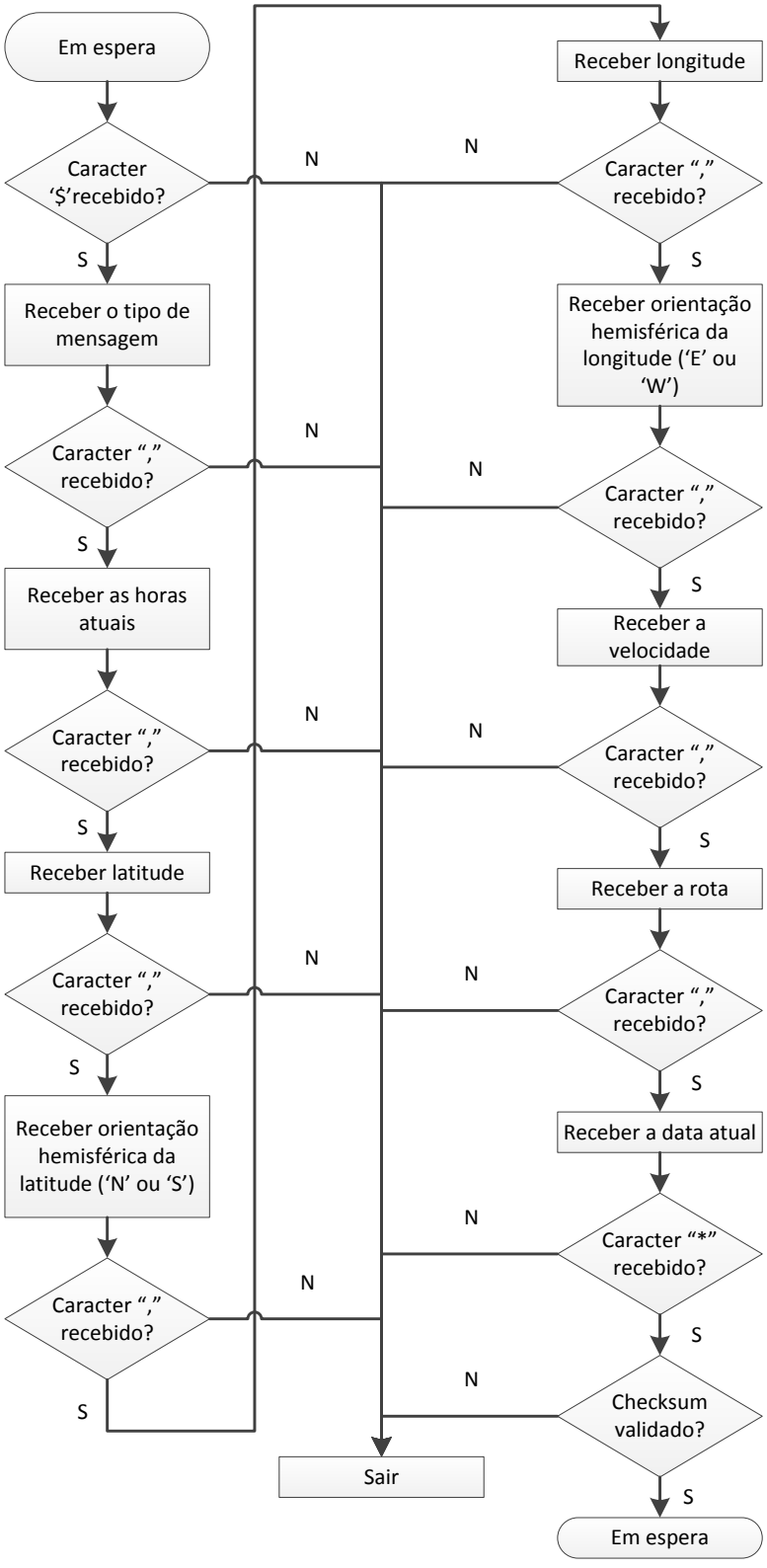

Figura 4.5 Diagrama de receção de mensagens NMEA GPRMC

4.3.1 Fórmula de *Haversine*

Através da Fórmula de Haversine [46], é então calculado o ângulo de direção entre ambas as localizações, através do norte geográfico e a diferença de ângulo da direção entre a posição desejada e a posição atual, ilustrado na Figura 4.6. A altitude é referenciada diretamente pela mensagem NMEA no formato GPGGA, sendo a posição e a rota definidas pela mensagem NMEA no formato GPRMC.

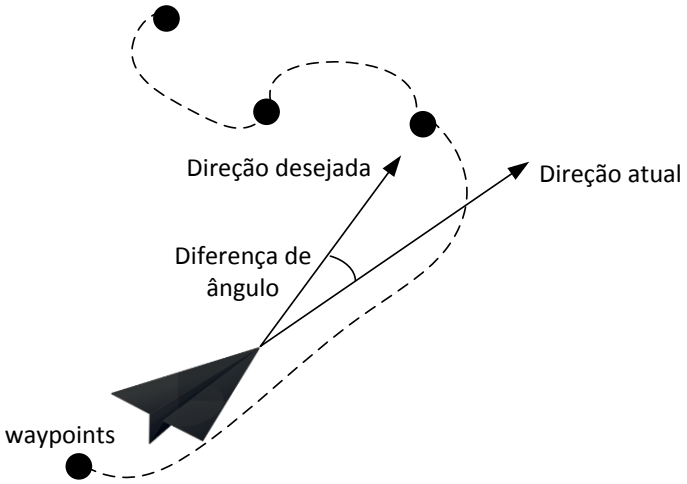

Figura 4.6 Diferença de ângulo entre direção desejada e atual

Para além do cálculo da diferença de ângulo é ainda determinada a diferença entre a altitude pretendida e a altitude atual.

Utilizando a fórmula de *Haversine* é possível, através das equações 4.1, 4.2 e 4.3, calcular a distância entre dois *waypoints* que, combinada com a diferença entre a direção atual e o posicionamento absoluto do Norte, dá origem à diferença de ângulo entre rota desejada e rota atual.

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \times \cos\varphi_2 \times \sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{4.1}$$

$$c = 2 \times atan2(\sqrt{a} \times \sqrt{(1-a)}) \tag{4.2}$$

$$d = R \times c \tag{4.3}$$

Onde:

- R: Raio da Terra (R = 6371 Km);
- φ: Latitude;
- λ : Longitude.
- a: Quadrado de metade da distância entre os WP;
- c: Distância angular em radiados;

- d: distância geográfica entre pontos.

Em que a função atan2 permite a devolução do quadrante do ângulo entre os dois pontos (y, x).

Para calcular a diferença de ângulo entre as duas posições:

$$y = \sin(\Delta\lambda) \times \cos\varphi_2 \tag{4.4}$$

$$x = \cos\varphi_1 \times \sin\varphi_2 - \sin\varphi_1 \times \cos\varphi_2 \times \cos(\Delta\lambda) \tag{4.5}$$

$$\theta = atan2(y, x) \tag{4.6}$$

A partir das equações 4.4, 4.5 e 4.6 é possível obter o ângulo de rotação θ necessário para o *drone* se direcionar em linha reta para a posição destino.

4.3.2 Posicionamento

O processo de posicionamento descrito na Figura 4.7 engloba as fases da aquisição de coordenadas GPS.

Numa primeira fase são enviadas as mensagens recebidas em tempo real pelo recetor de GPS.

De seguida, as mensagens de posicionamento são convertidas em coordenadas no formato de graus decimais, realizando-se um processo de leitura de duas mensagens válidas. Caso o processo seja concretizado com sucesso, é executada a identificação da posição atual até que seja atingido o objetivo, caso contrário realiza-se o processo anterior novamente.

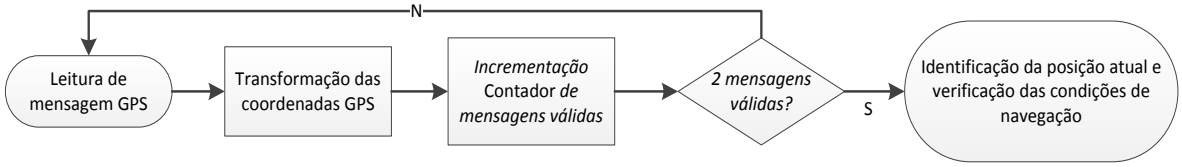

Figura 4.7 Ciclo de leitura e descodificação das coordenadas GPS

Quando identificada a posição atual, o procedimento prossegue para a verificação do estado atual de navegação que, entre os existentes, poderão ser:

- Início de navegação e descolagem;
- Atualização da posição em voo;
- Atualização das coordenadas do próximo WP (quando atingido o WP anterior);
- Atualização das coordenadas do próximo WP *home* (no caso de não existirem mais WP a alcançar);
- Aterragem e término da navegação.

.

Capítulo 5

Implementação do Sistema

Neste capítulo são descritos os algoritmos desenvolvidos associados ao sistema de navegação autónomo.

Será detalhado inicialmente o modelo de execução do controlador interno do *drone* para uma melhor perceção do controlo realizado. De seguida, é exposto o mecanismo de navegação autónomo proposto.

São ainda descritos os procedimentos de acondicionamento do *hardware* associado ao sistema. Em anexo são documentados detalhes técnicos à arquitetura física estipulada para o funcionamento do sistema.

5.1 Sistema de Controlo

O sistema-base de controlo do *drone* é responsável pela configuração, comunicação e controlo. Este será carregado para o microprocessador interno por FTP através de uma ligação gerada pelo protocolo de ligação telnet, onde será posteriormente corrido através do *arduino*.

O sistema está subdividido em várias funções. Na função principal (*ITDrone.c*) são geradas as configurações essenciais de comunicação e controlo do *drone*.

5.1.1 Visão Geral do Sistema

A estrutura da função principal é descrita na Figura 5.1.

Associados à função principal, estão vários ficheiros que contêm múltiplas funções de suporte.

Na função principal inicia-se o processo de comunicação, bem como os parâmetros associados à comunicação *wireless*, sendo os respetivos comandos enviados através de pacotes UDP. São ainda definidos variáveis globais afetas ao sistema.

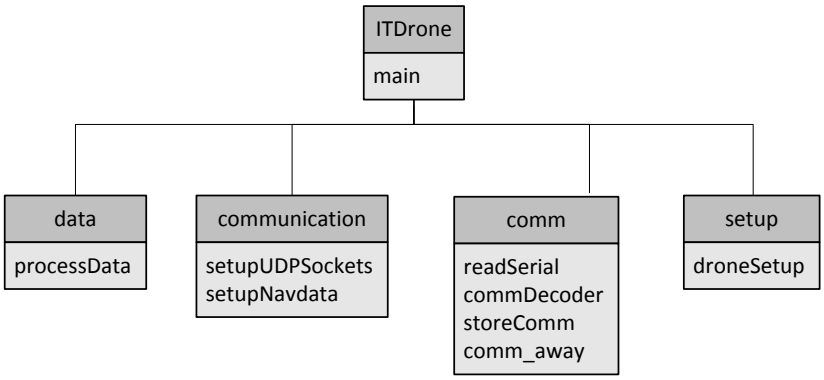

Figura 5.1 Estrutura de funções adjacentes à rotina principal *main* do ficheiro *ITDrone*

O resumo do funcionamento do sistema é detalhado na Figura 5.2.

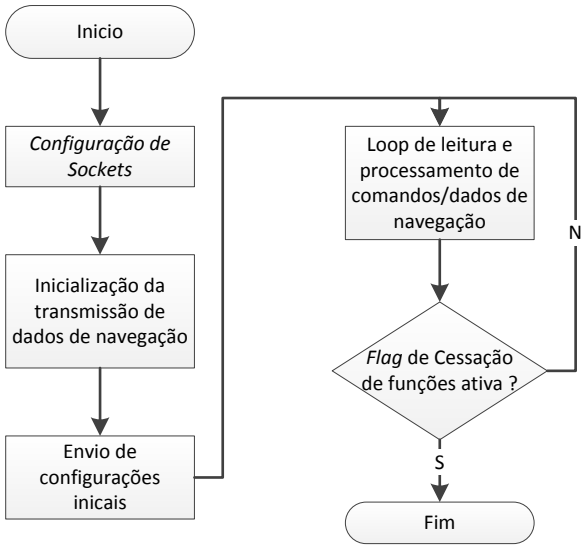

Figura 5.2 Estrutura base do sistema de controlo

5.1.1.1 Configuração da Transmissão de Pacotes via UDP

O primeiro passo restringe-se à configuração de *sockets* na função "*setupUDPSockets*", presente no ficheiro "*communication.c*". Nesta função são gerados e configurados os *sockets* necessários para a efetuar a transferência de comandos e receção de dados de navegação através de pacotes UDP entre o *drone* e o terminal.

5.1.1.2 Inicialização da Transferência de Dados de Navegação

De seguida, ainda no mesmo ficheiro, invocar-se-á a função "*setupNavdata*".

A partir desta função, é encaminhada uma mensagem de navegação através do *socket*, responsável pela transferência de dados de navegação, dando início ao processo de transmissão na Figura 3.9. Em caso de sucesso é ativada a *flag* "*navdataEnable*".

5.1.1.3 Configuração Inicial

Após a iniciação com sucesso da transmissão de dados de navegação, é invocada a função "*droneSetup*", onde se aguarda que o processo de transmissão de *navdata* seja concluído e, consequentemente, sejam enviados pelo *drone* dados de navegação continuamente. Será gerado um *timeout*, caso não seja processada *navdata* válida durante 500 milissegundos após o início de transmissão. A função responsável pela transferência de informação, entre o utilizador e o *drone*, é a função "*processData*" presente no ficheiro "*data.c*" descrita com mais detalhe no sub-capítulo 5.1.2. Concluído o processo, iniciar-se-á a transmissão das configurações iniciais previamente descritas (sub-capítulo 3.6.3).

Após a conclusão da inserção dos comandos no *buffer*, estes serão processados e transferidos para o *drone*, através da função "*processData*".

Main Loop

Concluído o processo de configurações iniciais, dar-se-á início ao *main loop* descrito na Figura 5.3.

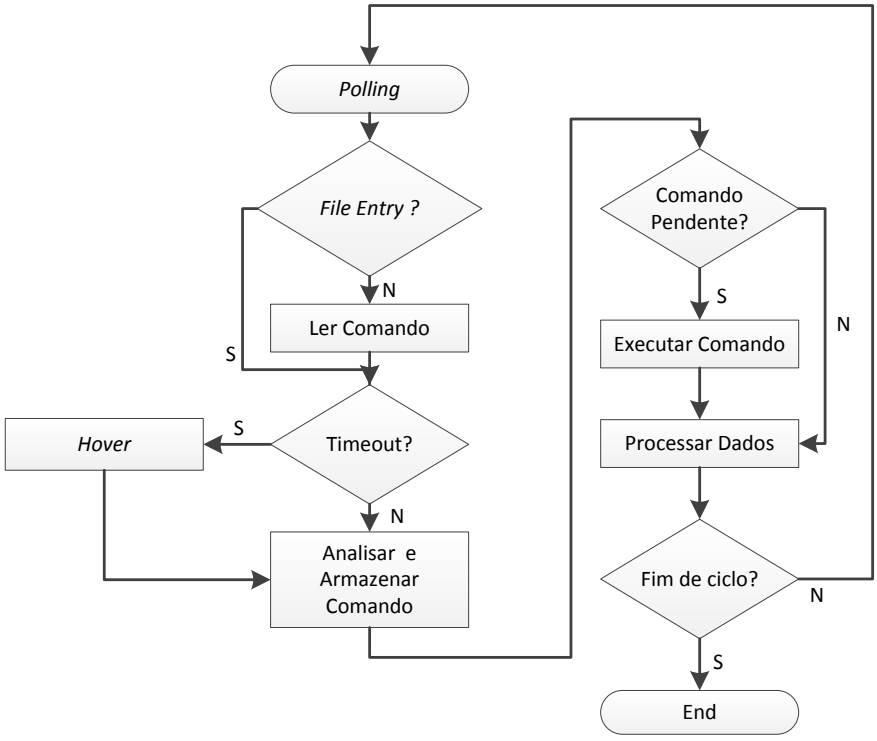

Figura 5.3 *Main loop* da função principal

Como primeira operação, é executado um ciclo de *polling* até que seja recebido um comando ou que ocorra um *timeout* (após 20 ms do início do ciclo). Após receção do comando, o tempo da receção (*io_lastTimeUpdate*) é atualizado com o tempo atual.

Leitura de comandos

Terminado o processo de *polling* e na condição de não ter sido gerado ou recebido qualquer ficheiro, é invocada a função "*readSerial"* presente no ficheiro "*comm.c*" onde é lido e armazenado o comando recebido via terminal, no respetivo *buffer* de comandos, caso este ainda não esteja cheio. É então efetuada uma verificação de *timeout* tal como relatado no sub-capítulo 3.6.1, verificando-se a existência (ou não) de novos comandos durante um período de 20 milissegundos.

Caso não seja recebido qualquer comando durante os 2 segundos, desde a última atualização do tempo de receção, é invocada a função "*away*" presente no ficheiro "*comm.c*", que efetua um *reset* aos parâmetros de movimento do *drone*, provocando um estado de *hovering* no *drone* (caso este esteja em voo).

Análise e armazenamento de comandos recebidos

Posteriormente, é invocada a função "*storeComm*" onde é analisado o tipo de comando recebido (caso este tenha sido armazenado no *buffer*); é verificada a validade do comando recebido através da inserção (ou não) do caráter iniciador '$'. Verifica-se ainda a validade da dimensão do comando de acordo com o tamanho máximo definido e, caso se confirme, este é copiado para o *buffer* de comandos a enviar.

Descodificação dos comandos recebidos

Caso exista algum comando pendente para ser descodificado, será executada a função "*commDecoder*" onde é efetuada uma verificação de correspondência do comando recebido e os comandos de controlo existentes e, caso essa exista, os argumentos do comando serão atualizados. Uma descrição mais detalhada é realizada no sub-capítulo 5.1.3.

5.1.2 Processamento de Comandos e Dados de Navegação

É na rotina de processamento de comandos e dados de navegação que todos os comandos e argumentos são atualizados e enviados para o *drone*, tal como são recebidos os dados de navegação e posteriormente descodificados. Através de um processo de *polling* são enviados os comandos ao mesmo tempo que os dados de navegação são enviados.

Todo o processo de descodificação de dados recebidos e codificação de dados a enviar é efetuado no controlador externo.

Na Figura 5.4, apresenta-se a estrutura das funções adjacentes à função de processamento de dados e controlos (*processData*).

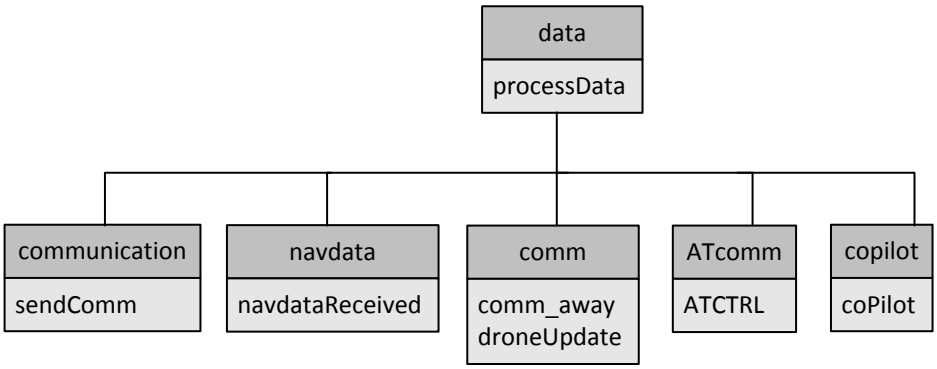

Figura 5.4 Estrutura das funções adjacentes à função *processData*

O procedimento realizado na função é descrito na Figura 5.5.

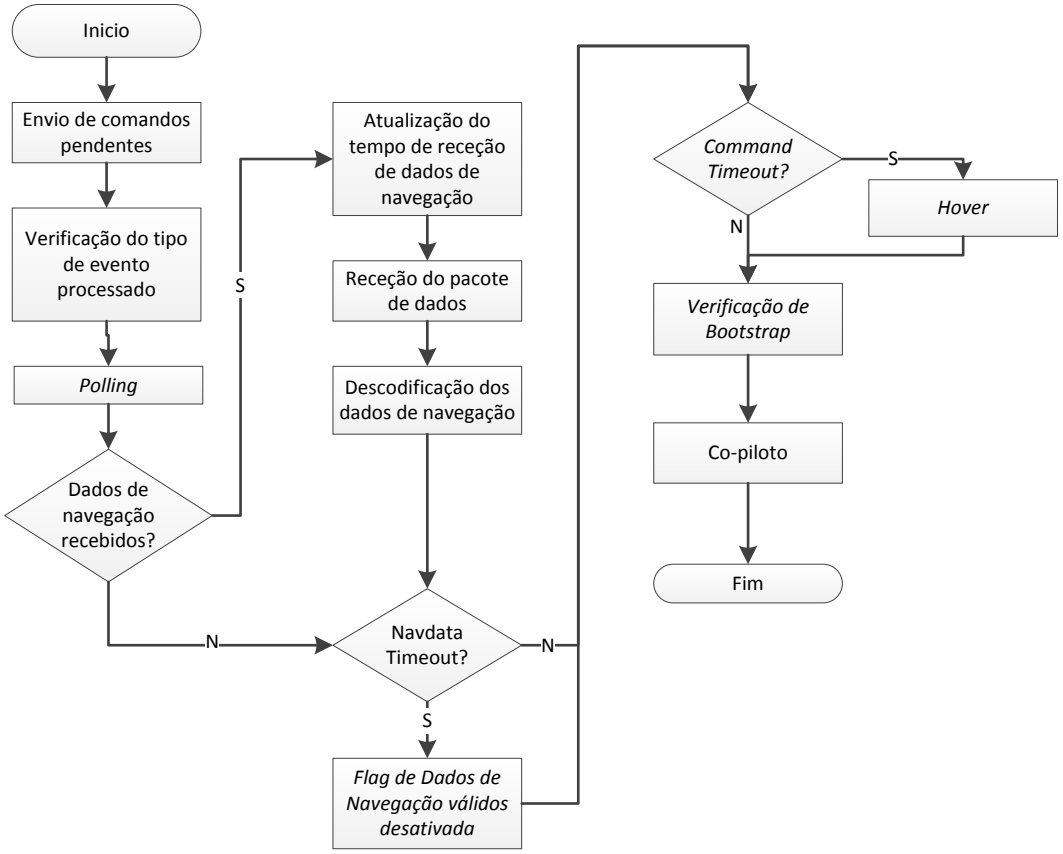

Figura 5.5 Processo de descodificação de dados de navegação e de envio de comandos

Inicialmente e na condição de o *buffer* de comandos não estar vazio, é enviado o seu conteúdo através da função *sendComm*. Recorrendo à função contida no ficheiro *communication.c*, é enviado o conteúdo inserido no *buffer* através do *socket*, responsável pelo envio de comandos através do endereço IP definido para o recetor.

De seguida, é efetuada a verificação das *flags* identificadoras do tipo de dados a receber. Caso alguma destas esteja ativada, é atribuído ao ficheiro uma descrição e associado ao tipo de evento seguinte, um ciclo de *polling*.

5.1.2.1 *Polling*

Concluído o processo de *polling*, é verificado qual o tipo de evento retornado. Caso a descrição do ficheiro seja coincidente com a descrição do *socket* de *navdata*, é atualizado o tempo da última receção de *navdata* para o atual, sendo posteriormente recebido um pacote de dados proveniente do *socket* dedicado a *navdata*, diretamente inserido no *buffer* associado ao *socket* de receção de dados de navegação, sendo salvaguardado também o tamanho em *bytes* do pacote recebido. Após a receção do pacote de dados, este será para a função *navdataReceived* descrita no sub-capítulo 3.6.2, onde todos os dados de navegação recebidos serão descompactados e descodificados.

5.1.2.2 Processamento de *Timeout*

Tal como na função principal *main*, é efetuada uma verificação de *timeout*, tanto para a *navdata*, como para os comandos, entrando o *drone* em modo *hovering* caso nenhum comando seja recebido durante um período de 2 segundos.

Adicionalmente, à margem do procedimento descrito na Figura 3.9, é verificado o estado de *status_bit_mask*. O processo é descrito na Figura 5.6.

De seguida, processam-se os automatismos do *drone*, nomeadamente da atualização do ângulo de direção e da altura configurada, na função *coPilot*, presente no ficheiro *copilot.c*. São também verificados estados como a percentagem de bateria, o controlo de altitude ou a aterragem de emergência, sendo considerados como processos de segurança e de controlo do *drone*. A função será detalhada no sub-capítulo 5.1.5.

Concluídas as verificações dos estados do *drone*, tendo estas sido positivas, é enviado um pacote de dados contendo os comandos de controlo e respetivos parâmetros atualizados através da função *droneUpdate*, previamente descrita.

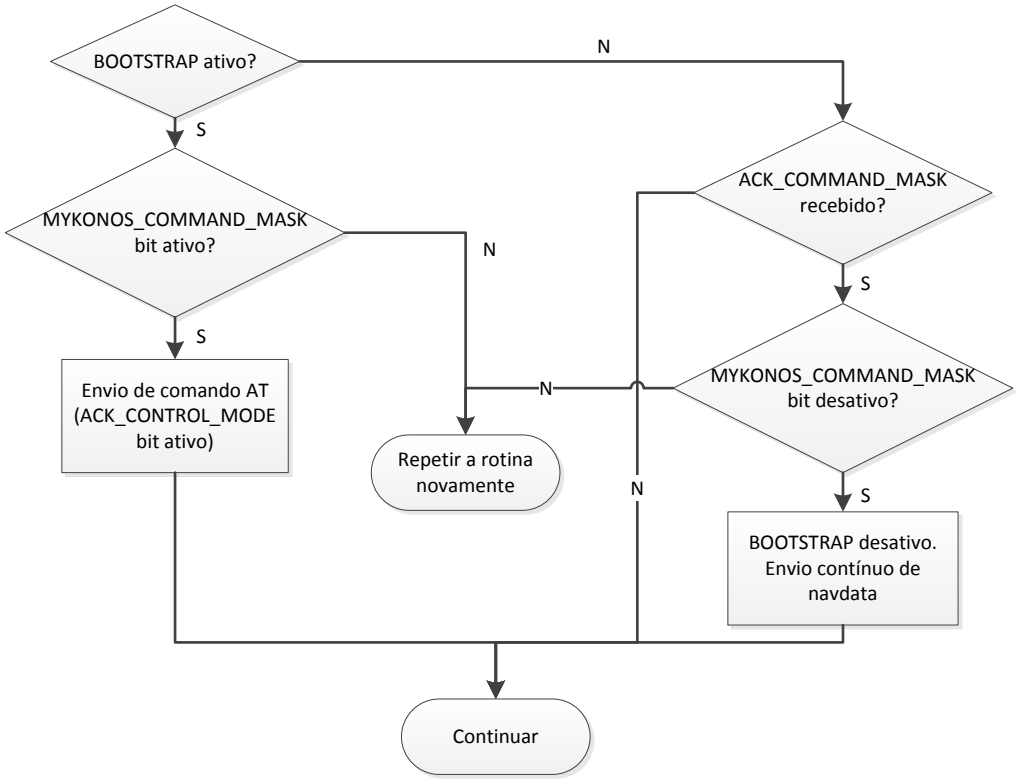

Figura 5.6 Verificação do *status_bit_mask* (bit correspondente ao processo de *BOOSTRAP*)

5.1.3 Comandos de Controlo

A introdução de comandos de controlo tem que, como já anteriormente referido, passar por um processo de receção, análise e descodificação. Estes três processos estão separados nas funções *readSerial*, *storeComm* e *commDecoder*, respetivamente. Como suporte da função *commDecoder*, onde é analisado qual o comando a executar dependendo do tipo de dados presentes no *buffer* de comandos, existem funções auxiliares de suporte, referentes a cada tipo de comando existente. A estrutura dos comandos de controlo é então apresentada na Figura 5.7.

Inicialmente, na função de descodificação do comando é verificado o número e tipo (inteiro ou *float*) de argumentos que o comando apresenta. Depois, existindo uma correspondência do comando e tipo de argumentos que o comando apresenta com os comandos existentes, é efetuada uma chamada da função correspondente ao comando em questão.

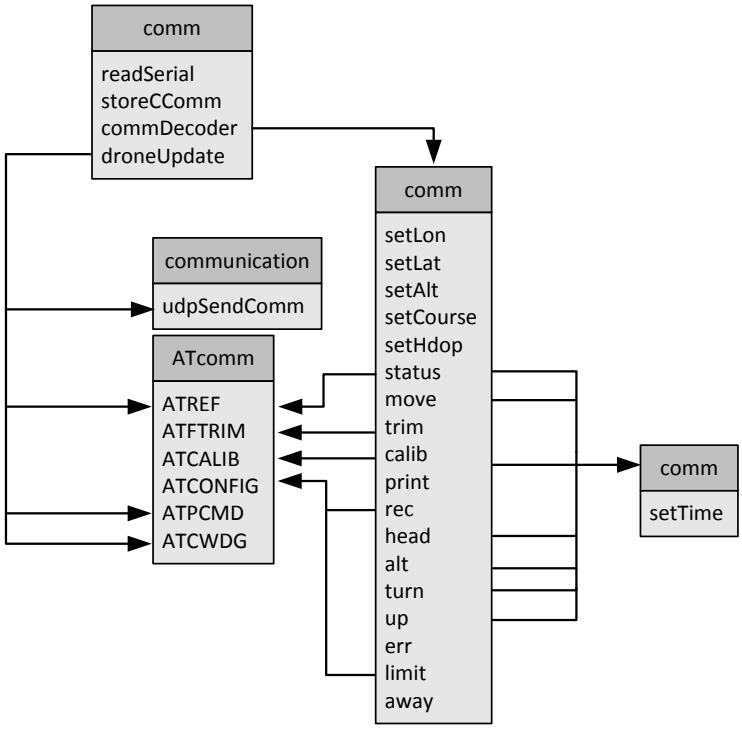

Figura 5.7 Estrutura dos comandos de controlo do *drone*

5.1.3.1 Comandos

Cada comando executa tarefas distintas, entre as quais:

- ***setLon***:
    - Define o valor de longitude lido;
- ***setLat***:
    - Define o valor de latitude lido;
- ***setAlt***:
    - Define o valor de altitude lido;
- ***setCourse***:
    - Define o valor do ângulo de direção lido;
- ***setHdop***:
    - Define o valor de HDOP lido;
- ***status***:
    - Atualiza as *flags* de controlo do *drone* consoante o seu estado:
        - 0 (desligado): desativa as *flags* de voo, altitude e direção;
        - 1 (Voar): ativa a *flag* de voo;

        - 666 (Ativar Emergência): Além de adicionar a sequência de comandos referentes ao estado de emergência, desativa as *flags* de voo, altitude e direção, realizando ainda um *reset* ao número de sequência;
        - 777 (Desativar Emergência): Adiciona o comando referente à desativação do estado de emergência e desativa as *flags* de voo, altitude e direção, possibilitando o *drone* de voltar a voar quando assim o piloto entender;
    - Atualiza o tempo de execução de comando de controlo (*setTime*);
- ***move***:
    - Atualiza os parâmetros de controlo do *drone* (*roll*, *pitch*, *gaz*, *yaw*);
    - Desativa as *flags* de controlo automático caso seja introduzida uma aceleração vertical;
    - Desativa a *flag* de piloto automático caso seja introduzida uma aceleração rotacional;
    - Altera o modo de voo para receção de comandos progressivos;
    - Atualiza o tempo de execução de comando de controlo (*setTime*);
- ***trim***:
    - Adiciona um comando referente à calibração dos controladores internos;
- ***hover***:
    - Iguala os argumentos do comando de voo a zero, bem como as *flags* de controlo de altitude e direção, indicando ao *drone* para pairar no ar;
- ***calib***:
    - Adiciona o comando referente à calibração do magnetómetro e atualiza o tempo de execução de comando de controlo (*setTime*);
- ***print***:
    - Imprime os estados atuais do *drone*, nomeadamente:
        - Altitude;
        - Percentagem de bateria;
        - Estado de controlo;
        - Ângulo de *roll*;
        - Ângulo de *yaw*;
        - Ângulo de *pitch*;
        - Tempo de atualização de *navdata*;
        - Erro de ângulo de direção;
        - Erro de altitude.

- ***head***:
    - Define o ângulo de rotação e ativa a correspondente *flag* (*call* da função *coPilot*);
    - Atualiza o tempo de execução de comando de controlo (*setTime*);
- ***alt***:
    - Define a altitude e ativa a referente *flag* (*call* da função *coPilot*););
    - Ativa a *flag* de auto-controlo (*call* da função *coPilot*););
    - Atualiza o tempo de execução de comando de controlo (*setTime*);
- ***turn***:
    - Atualiza o valor do ângulo de direção do *drone* e ativa a referente *flag*;
    - Atualiza o tempo de execução de comando de controlo (*setTime*);
- ***up***:
    - Atualiza o valor da altitude e a sua referente *flag*;
    - Atualiza o tempo de execução de comando de controlo (*setTime*);
- ***err***:
    - Atualiza os valores de erro de altitude e direção;
- ***limit***:
    - Define os valores de configuração (inclinação máxima, aceleração vertical máxima, ângulo de controlo máximo e altitude máxima);
    - Envia os comandos referentes às configurações enumeradas;
- ***away***:
    - Define os argumentos de movimento do *drone* como *off*, deixando o *drone* no estado de *hovering*;
- ***setTime***:
    - Recebe a hora atual e atualiza o tempo de execução de comando de controlo.

5.1.3.2 Envio de Comandos

Para o envio de comandos é executada a função *droneUpdate*.

Inicialmente, é feita a verificação do estado de voo do *drone* e, caso este se encontre em voo, é adicionado ao *buffer* de comandos o comando de controlo responsável pelo movimento (*AT*PCMD*), com os respetivos argumentos atualizados.

Posteriormente, é adicionado um comando de controlo referente ao *watchdog* (AT*CMWDG), provocando o *reset* do *timer*. É, de seguida, adicionado um comando referente ao estado de voo.

Após a inclusão de todos os comandos a executar no *buffer*, é enviado o seu conteúdo através da função *sendComm*, já anteriormente detalhada.

5.1.4 Dados de Navegação

A receção de dados de navegação é um processo essencial na avaliação dos estados do *drone* em tempo real, contribuindo para a posterior atuação dos controladores como exemplificado na Figura 4.4. Como tal, é definida uma função (*navdataReceived*) de descodificação e análise dos estados recebidos. A execução da função está inteiramente dependente do parâmetro *header* do pacote de dados de navegação, sendo que em caso de não existir uma correspondência entre o recebido e o espectável, o procedimento não é executado.

Inicialmente, em caso de correspondência é verificado se o bit de máscara do *watchdog* (*MYKONOS_COM_WATCHDOG_MASK*) está ativo. Caso esteja, significa que existe um problema na comunicação, pelo que é enviado um alerta efetuando-se de seguida um *reset* ao número de sequência (e consequente *reset* ao sistema).

No caso de o sistema estar a funcionar corretamente, é verificado se o número de sequência é superior ao valor da execução anterior e, em caso afirmativo, analisa se o *drone* se encontra num processo de envio de dados de navegação essenciais (*navdata demo*). Nesta situação, é executada a função de descodificação de pacotes de *navdata* (*navdataDecoder*) e calculado o *checksum* do pacote recebido. Caso o *checksum* seja válido, o pacote de dados é validado e o valor de sequência é atualizado.

5.1.4.1 Descodificação de Pacotes de *Navdata*

No seguimento da descrição de um pacote de dados de navegação na Figura 3.9, são atualizados os campos referentes ao pacote de dados, nomeadamente o *drone state* e o *vision flag*.

De seguida, verifica-se o valor do ponteiro referente aos blocos de dados. Caso este não seja nulo, averigua-se a validade da opção através do argumento *size*. Caso esta seja zero, os dados são inválidos e, portanto, é atualizado o valor do ponteiro para *NULL*; caso contrário, é identificado o tipo de dados inseridos no pacote através do *tag* dos blocos de dados, sendo estes posteriormente copiados para a variável correspondente do pacote de dados descodificado.

5.1.5 Co-piloto

A função de copiloto inserida no ficheiro *copilot.c* tem como missão, a de ajustar automaticamente a altura e a direção do *drone*, consoante a ativação das *flags* correspondentes. São ainda controlados os estados do *drone*, como já anteriormente referido. O procedimento de ajuste de direção e altura são ilustrados na Figura 5.8, respetivamente, sendo o ajuste efetuado de forma análoga em ambas as variáveis.

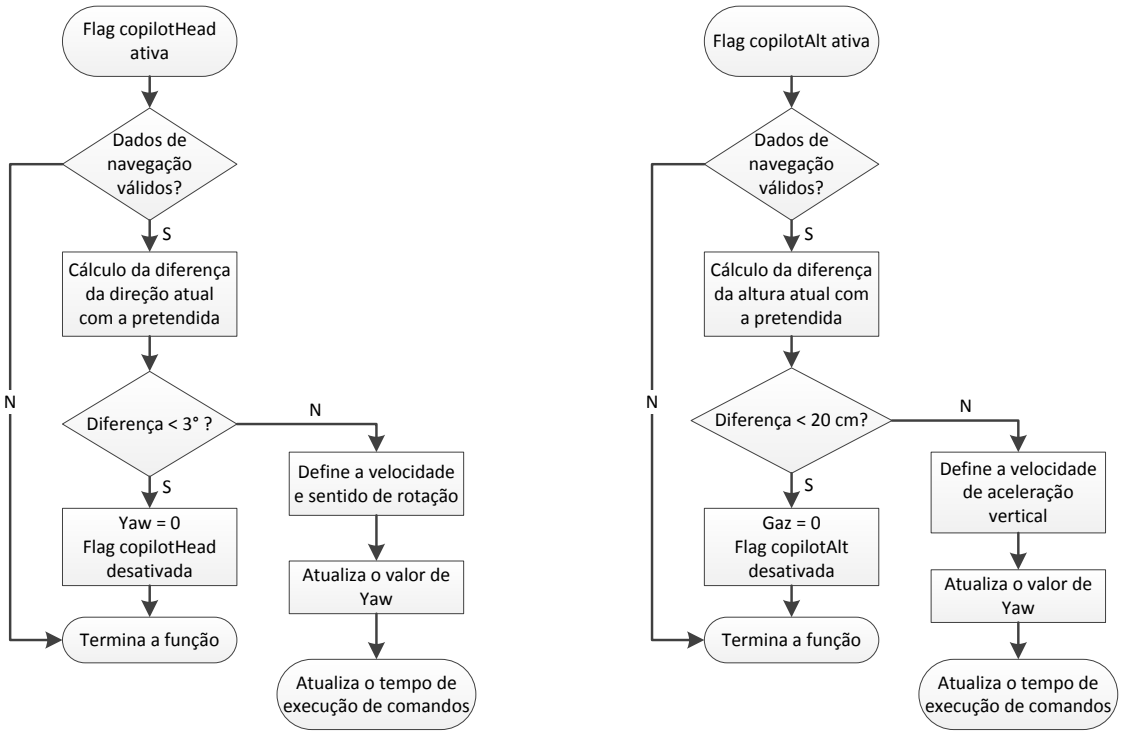

Figura 5.8 Processo de ajuste de direção

O co-piloto irá também, como já referido, auxiliar os controladores do *drone*, isto é, tomar decisões em casos específicos, entre os quais:

- Percentagem de bateria: Quando a percentagem está abaixo dos 10% e na condição de o *drone* se estar a voar a uma altura superior a 5 metros, esta é reduzida para os 3 metros; Quando a percentagem é inferior a 5%, este irá aterrar de forma automática;
- Voos a uma altura diminuta: No caso de o *drone* se encontrar a voar a uma altura inferior a 0.7 metros, este é automaticamente elevado para 1 metro, assegurando uma altura mínima que evite o choque com qualquer objeto que esteja no solo, durante o processo de voo;
- Modo de *hovering*: Quando é ativado o modo de *hovering*, com o auxílio da variável de erro de altitude, é controlada a altitude do *drone* por forma a que este permaneça o mais estável possível a uma altitude de 0.5 metros;
- Modo de emergência: Caso o *drone* entre em modo de emergência (i.e., o bit de *MYKONOS_ EMERGENCY_MASK* seja ativado) é atualizado o tempo de execução da última emergência efetuada, como também é ativada a *flag* correspondente ao estado de emergência. Caso a *flag* se encontre ativa e o modo de emergência seja desativado, a *flag* é igualmente desativada.

5.2 Sistema de Navegação Autónomo

O sistema de navegação autónomo é composto por uma função principal repartida entre as configurações iniciais e um *loop* de execução da navegação. Paralelamente existem várias funções que fornecem um auxílio ao processo de execução do sistema, tanto na configuração, como na execução.

A estrutura do sistema é ilustrada na Figura 5.9.

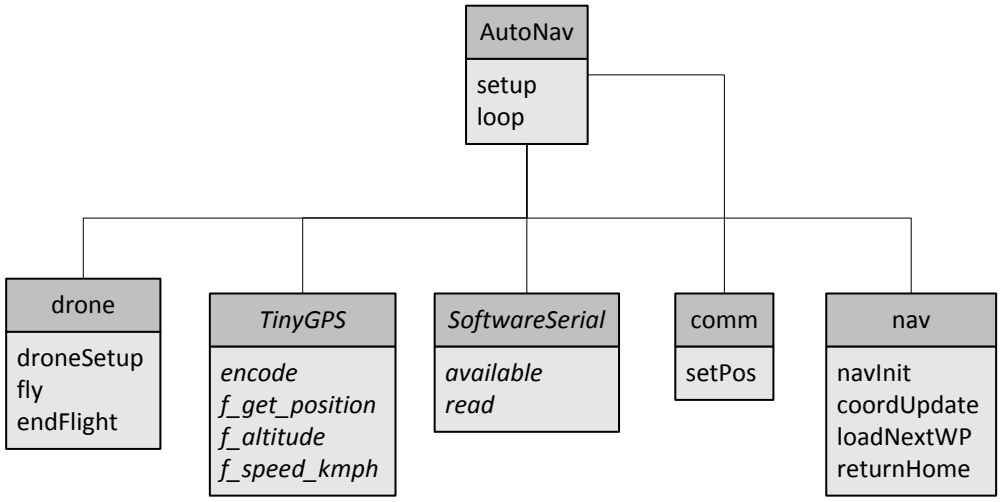

Figura 5.9 Estrutura das funções adjacentes ao sistema de navegação autónomo

Primeiramente, e tal como o próprio nome indica, na função *setup* irão ser realizadas as configurações iniciais. Começa-se por definir a taxa de transmissão de dados (*baudrate*) do *arduino* para o *drone* (115200) e para o recetor de GPS (9600). É ainda definida a frequência de transmissão de mensagens GPS a 5 Hz e realizada a filtragem de mensagens apenas para receção dos formatos GPGGA e GPRMC.

Depois, é executada a função *droneSetup* que, como o nome indica, irá enviar as configurações de controlo iniciais:

- Salvaguarda o ficheiro de controlo transferido para o *drone*;
- Configura as opções iniciais do *drone* como referido no anexo A.

Concluído o processo de configurações iniciais, é inicializado o *loop* de execução do sistema de navegação.

O processo realizado é descrito na Figura 5.10.

Ab initio, é analisado o número de bytes disponíveis no *buffer* de receção de coordenadas.

Caso existam bytes disponíveis, a primeira operação será a de receber um caráter através do porto RX e processá-lo na função *encode* da biblioteca *TinyGPS*.

O processo é repetido até que seja devolvida pela função *encode* uma mensagem válida de GPS, ligando o LED disponível no porto 13 do *arduino*, como sinalizador de mensagem válida recebida.

Posteriormente, são salvaguardadas as coordenadas da posição atual. Na eventualidade de serem as primeiras coordenadas recebidas estas serão salvaguardadas como as coordenadas mais recentemente validadas. Caso contrário, é determinada a distância entre as coordenadas atuais e as recentemente validadas e, no caso de a distância ser superior a 1 m, então a direção entre as posições será atualizada e a *flag* de controlo de direção é ativada. Por fim, a posição mais recentemente validada é atualizada com a atual.

O processo utilizado no cálculo da distância e direção entre coordenadas corresponde ao método de determinação da distância e direção através das fórmulas de *Haversine*, descrita no sub-capítulo 4.3.1. Este método é implementado nas funções *distance_between* e *course_to* presentes na biblioteca TinyGPS.

Concluído o processo de atualização de coordenadas e de ajuste de direção, através da função *posUpload*, é transmitida via série a atualização das coordenadas GPS e do valor de ângulo de rotação da rota a seguir, bem como o valor de HDOP.

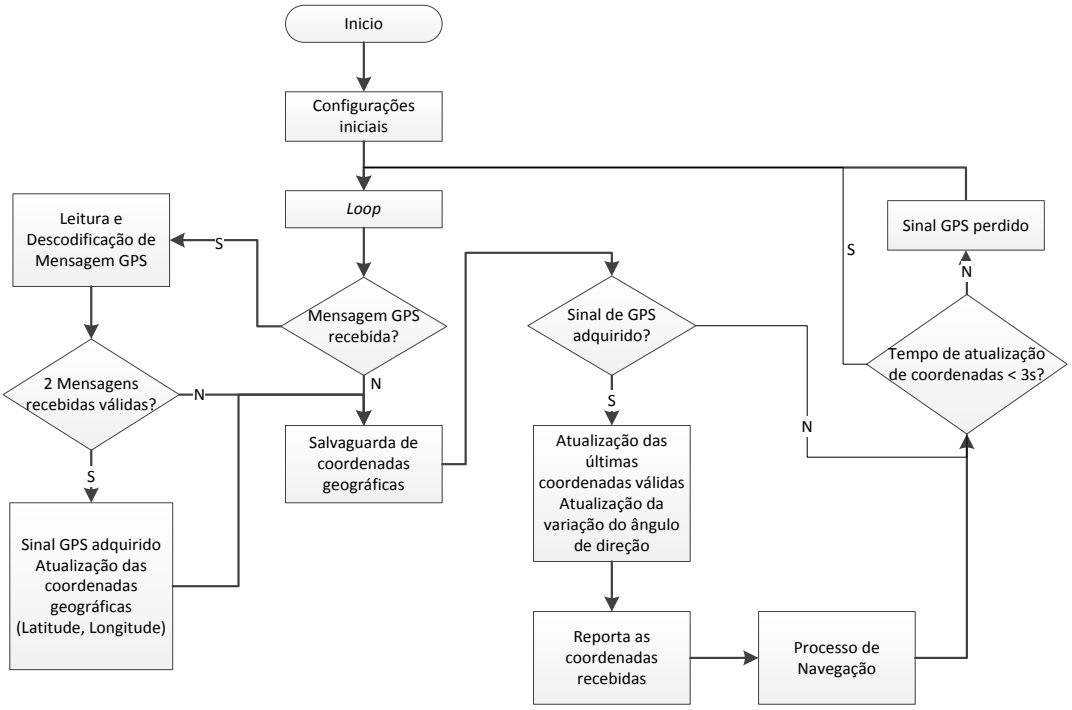

Figura 5.10 Processo do Sistema de Navegação Autónoma

De seguida, realiza-se o processo de navegação, que é ilustrado na Figura 5.11.

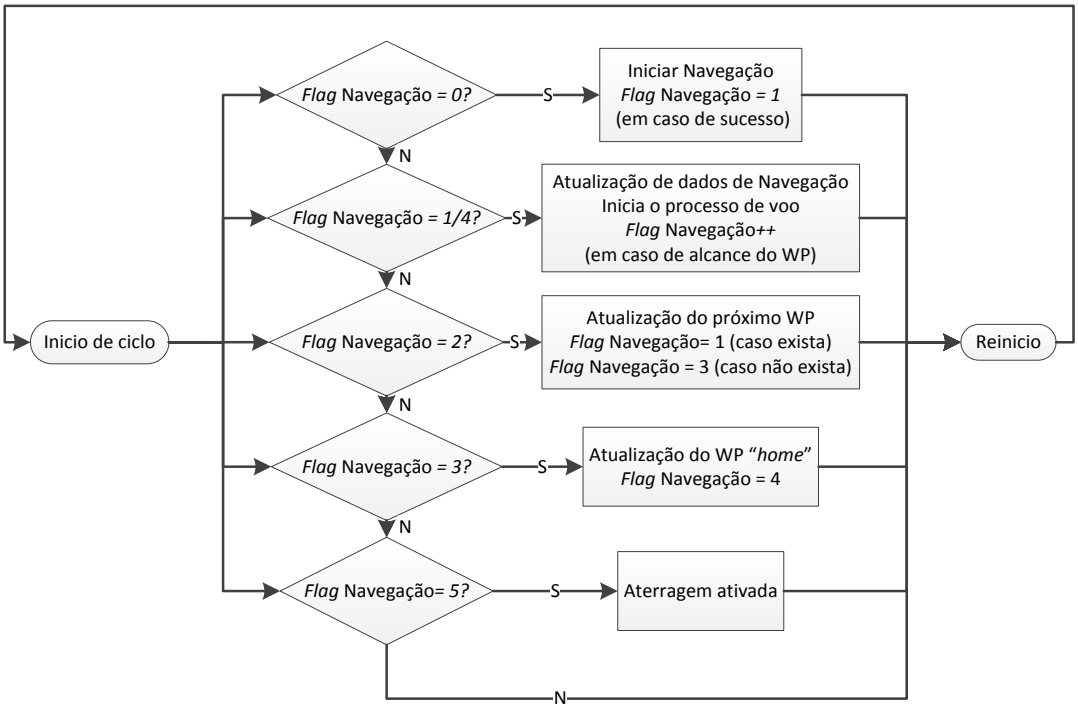

Figura 5.11 Processo de navegação autónoma

5.2.1 Iniciação de Navegação

O processo inicia-se com uma verificação do valor de HDOP e, caso este seja superior a 2 metros, a navegação é abortada; caso contrário, são salvaguardadas as coordenadas atuais de navegação como sendo as coordenadas da posição "*home*".

Depois, são carregadas as coordenadas do primeiro WP, sendo calculada a diferença de ângulo e a distância entre o WP e a posição "*home*" e caso esta seja superior a 20 metros, a navegação é interrompida. Caso contrário, é calculada a distância entre todos os WP introduzidos, sendo a navegação identicamente interrompida, no caso de a esta ser superior a 50 metros.

Concluídas as verificações de segurança, em caso de sucesso, é atualizada a direção da posição atual para a seguinte e, posteriormente, enviados os comandos de início de navegação com respetiva descolagem, calibração e definição de altitude máxima. Segue-se para o próximo passo do processo (Atualização dos dados de navegação).

5.2.2 Atualização dos Dados de Navegação

O processo inicia-se com a atualização quer da diferença angular entre a direção atual e a direção do WP a alcançar, quer da distância do ponto atual para o WP.

Seguidamente, serão atualizados os erros de leitura do *drone*, para posteriores correções.

É, então, iniciado o processo de voo, detalhado no sub-capítulo 5.2.4.

O processo irá repetir-se até que o WP seja alcançado (ou que a distância a este seja menor do que 1 metro) e, quando alcançado, a *flag* de navegação é incrementada, permitindo o carregamento do próximo WP.

5.2.3 Atualização do Próximo WP

Caso existam WPs por percorrer, as coordenadas do WP a alcançar serão atualizadas e a *flag* de navegação é decrementada, sendo realizado o processo anteriormente descrito.

Caso todos os WPs tenham sido percorridos, a *flag* de navegação é incrementada, e consequentemente, carregadas as coordenadas referentes à localização inicial "*home*".

Quando atingida a posição inicial (com margem de erro de 1 metro), é enviado o comando referente à aterragem e finalização de voo.

5.2.4 Processo de Voo

O processo de voo é um processo sensível, que exige uma análise cuidada sobre quais os estados reportados pelo *drone* e sobre o caminho deverá percorrer, isto é, deve assegurar-se que o *drone* prossegue o caminho corretamente definido, por forma a evitar acidentes de percurso.

Para tal, com uma verificação constante dos estados do *drone*, são definidos vários passos dentro do processo, entre os quais:

1. *Hover*;
2. Aguardar para que sejam recebidas as mensagens de *fix* do GPS;
3. Direcionar o *drone* para a posição do WP;
4. Voar em linha reta para a posição do WP;
5. Esperar que o posicionamento atual do *drone* seja atualizado;
6. Verificar a validade da rota a ser percorrida;
7. Direcionar o *drone* para o posicionamento do WP, caso esteja no corredor de navegação;
8. Voar em linha reta para o posicionamento do WP, caso esteja no corredor de navegação;
9. Corrigir o posicionamento por forma a entrar no correr de navegação.

A gestão dos passos a serem executados no processo de voo é descrita na Figura 5.12.

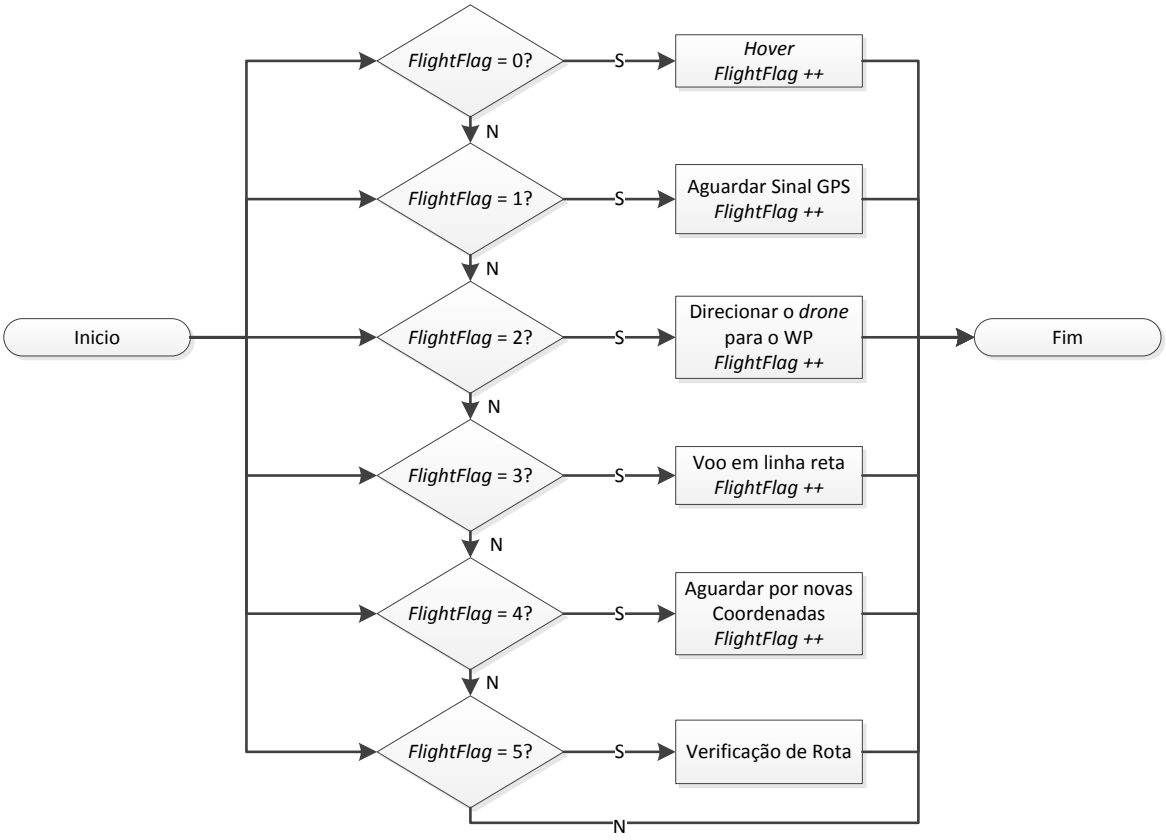

Figura 5.12 Gestão do processo de voo autónomo

5.2.4.1 *Hover*

Inicialmente, é transmitido um comando de *hovering* possibilitando ao *drone* e aos sensores internos realizarem um processo de estabilização.

A *flag* de próximo passo (*FlightFlag*) é então incrementada.

5.2.4.2 Aguardar Sinal de GPS

Nesta fase, far-se-á um compasso de espera, aguardando que o sinal de receção de GPS seja recebido de forma precisa.

A *flag* de próximo passo é então incrementada.

5.2.4.3 Direcionar o *Drone* para o WP

Antecedendo o movimento frontal do *drone*, é calculada a diferença angular entre a direção atual e a pretendida. Após a verificação do valor, é enviado o comando de rotação com o respetivo valor em graus.

O procedimento da função é ilustrado na Figura 5.13.

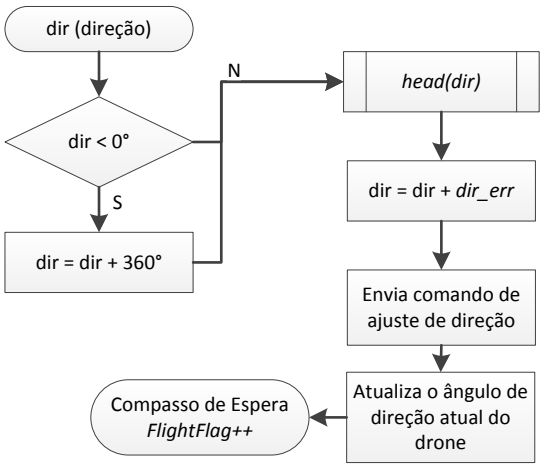

Figura 5.13 Rotina de direcionamento para o WP pretendido

5.2.4.4 Voo em Linha Reta

De forma a facilitar o processo de voo, a movimentação entre posições é efetuada sempre em linha reta, sendo o processo de averiguação de rota simplificado.

Preliminarmente, é atribuído um valor de inclinação frontal (*pitch*) para que seja realizado um movimento frontal e, posteriormente, é executada a função *move* encarregue de enviar o comando de movimento com os parâmetros de voo atualizados.

Tendo sido realizado a verificação do ângulo de direção, este é reiniciado e, finalmente, é incrementada a *flag* de próximo passo.

5.2.4.5 Aguardar Por Novos Dados de Navegação

Neste momento do processo é feito um compasso de espera até que seja recebida uma nova trama de coordenadas atualizadas da posição. Após receção das coordenadas verifica-se a variação na direção do WP pretendido.

Caso a direção ou a altura tenham sofrido alterações, são calculadas as diferenças de ângulo e altura e, posteriormente, corrigidas nos parâmetros do comando de movimento enviado através da função *move* anteriormente descrita. Em caso de receção de novos dados de navegação, a *flag* de próximo passo é incrementada.

Este passo previne possíveis desvios de direção ou altura provocados por elementos externos como por exemplo, o vento.

5.2.4.6 Verificação de Rota

Neste momento do processo é essencial verificar se o *drone* está próximo do objetivo (a menos de 2 metros de distância).

Caso não se verifique, ir-se-á averiguar se este se encontra no corredor de navegação, sendo calculada a distância entre o WP e o ponto a atingir atualmente, em função da distância e do erro de ângulo. Caso este esteja a mais de 2 metros de distância (lateral) ou 6 metros de distância (em linha reta) do objetivo, considera-se que o *drone* se encontra fora do corredor de navegação. Os passos seguintes estão obviamente dependentes da verificação da rota. Como tal, é esquematizado na Figura 5.14 a dependência entre os próximos passos e o atual.

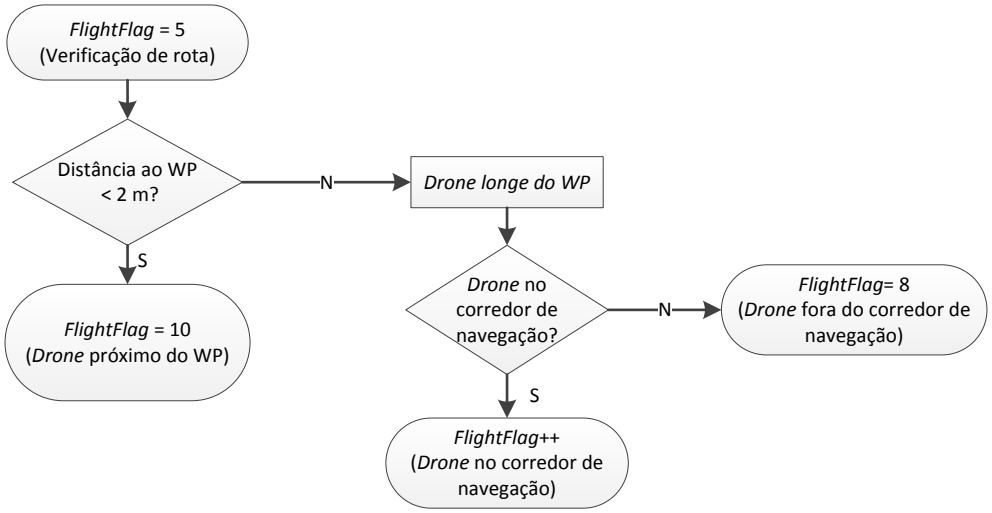

Figura 5.14 Processo de Verificação de Rota

5.2.4.7 *Drone* no Corredor de Navegação. Correção de Direção e Movimento Frontal

Na conjuntura atual é verificada a diferença entre a direção atual e a direção correta da rota a tomar. Caso seja diferente de zero, é efetuada uma correção de direção já ilustrada na Figura 5.13.

Feita a correção, a *flag* de passo seguinte é incrementada, sendo, posteriormente, transmitido o comando de movimento frontal em direção ao WP pretendido.

Neste passo é efetuada a correção de ângulo necessária para direcionar o *drone* no sentido do WP, decrementada a inclinação frontal para uma aproximação célere e uma diminuição para cerca de metade da inclinação lateral, fazendo com que a deslocação seja o mais retilínea possível.

É, então, invocada a função *move* responsável pelo envio do comando de movimento com os parâmetros previamente configurados. Após a execução do movimento, verifica-se se o *drone* continua no corredor de navegação; caso assim não aconteça, executa-se, novamente, o processo de verificação de rota anteriormente descrito.

5.2.4.8 *Drone* Fora do Corredor de Navegação: Correção de Direção e Movimento Frontal

Num processo análogo ao anterior é verificada a diferença de ângulo entre a direção atual do *drone* e a direção para o WP pretendido. Caso a diferença não seja nula, é, de novo, efetuada uma correção de direção já ilustrada na Figura 5.13.

É incrementada a *flag* de passo seguinte, passando para a configuração do comando de movimento. Na eventualidade de o *drone* se encontrar fora do corredor de navegação, a inclinação frontal é reduzida para metade, sendo importante dar prioridade à correção da direção e não ao movimento frontal. É então enviado o comando de movimento para a correção da posição atual do *drone*.

Concluído o processo, é verificado se o *drone* se encontra, finalmente, no corredor de navegação. Em caso afirmativo, é executado o passo 5, referente à verificação de rota.

5.2.4.9 *Drone* Próximo do Objetivo

Repetindo-se o processo anterior, analisa-se a diferença de ângulo entre a direção atual do *drone* e a direção do WP pretendido.

Incrementada a *flag*, prossegue-se para a realização do movimento frontal do *drone* aplicando uma velocidade consoante a distância ao WP (quanto mais próximo se encontrar, maior a redução de velocidade). No fim, é enviado o comando de movimento do *drone*.

5.2.5 Cessação de Funções

Após ter sido alcançada (com um erro de 1 metro de distância) a localização final, *home*¸ são então enviados os comandos referentes ao término do voo. Para tal, é executada a função *endFlight* encarregue de enviar os comandos necessários para que seja executado o processo de aterragem e interrupção dos motores.

5.3 Sistema Físico

O plano inicial de alimentação do sistema de controlo e o sistema de navegação (i.e., o plano de alimentação do *arduino*) seria o de tirar proveito da alimentação de 12V proveniente da bateria do *drone*, no entanto, e devido ao pouco tempo útil de voo que estas já fornecem, optou-se por alterar a alimentação para uma pilha de 9V (como está presente na Figura 5.15), sob pena de reduzir ainda mais o tempo de voo, fornecendo assim alimentação para o *arduino* e para o recetor GPS.

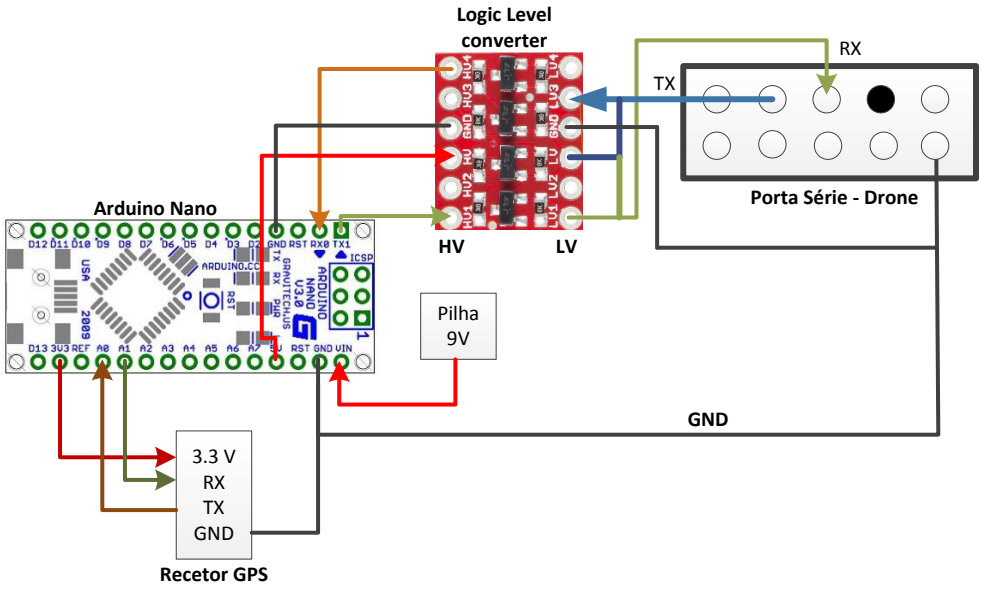

Figura 5.15 Ligações físicas entre o recetor GPS, *arduino* e *drone* [3] [47]

Posteriormente, é necessário realizar uma proteção entre as portas série do *drone* e do *arduino*, devido a estas transmitirem/receberem dados a 1.8V e 5V, respetivamente.

Como tal adquiriu-se um *logic level converter*, cujo esquemático está presente na Figura A.2. que é constituído por quatro vias de comunicação reguladas à tensão pretendida, através de quatro transístores BSS138 tipo-N [48]. A sua função será a de garantir que a comunicação seja realizada sem que seja necessário que ambos os dispositivos transmitam/recebam informação com a mesma alimentação.

O resultado da integração do sistema físico no drone está presente na Figura 5.16.

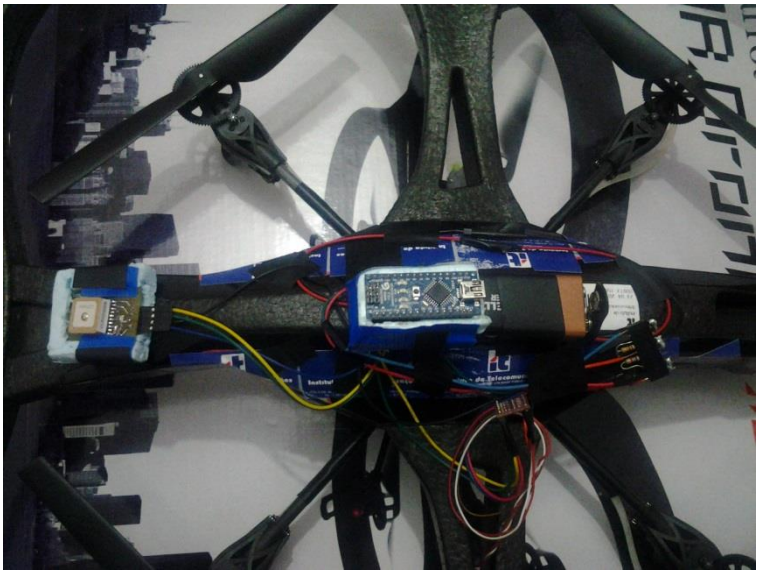

Figura 5.16 Montagem do sistema de navegação no *drone*

Pode ainda visualizar-se na e Figura 5.17 as ligações (TX, RX e GND) entre a porta série do *drone* e o *arduino*, que facilita a transmissão dos comandos AT.

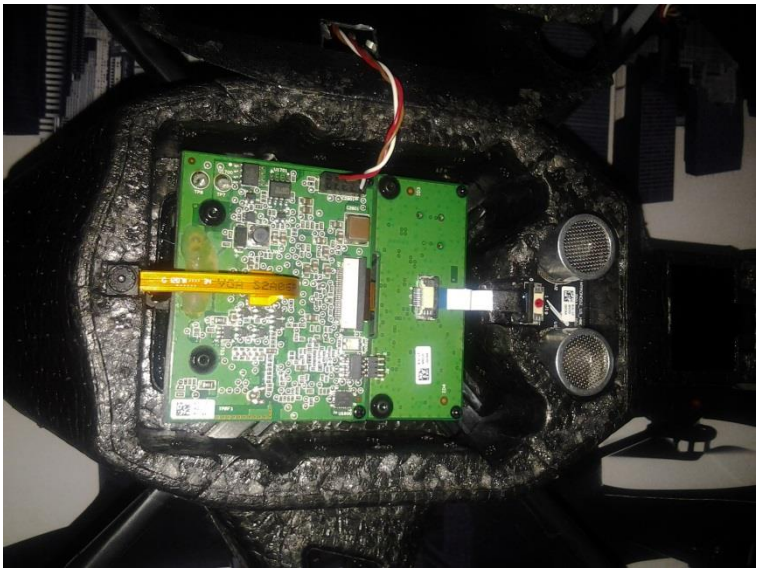

Figura 5.17 Ligação da porta série (TX, RX e GND) do *Drone* para o *Arduino*

5.4 Resultados

Em primeiro lugar, efetuaram-se testes de segurança e fiabilidade do sistema passo-a-passo, com o objetivo de evitar que na integração do sistema no *drone* e consequentes testes, este falhasse e existisse a possibilidade de queda e destruição parcial/total do sistema físico.

Em segundo lugar, realizaram-se testes ao algoritmo de controlo, como demonstrado na Figura 5.18, em que se mostra o algoritmo a ser executado no computador, com os comandos a serem introduzidos através do terminal e, posteriormente, encaminhados nos *sockets* UDP gerados e, previamente, detalhados. Fica ainda demonstrado na figura a devolução dos dados de navegação aquando do envio do comando executado na função *print*, descrita no sub-capítulo 5.1.3.

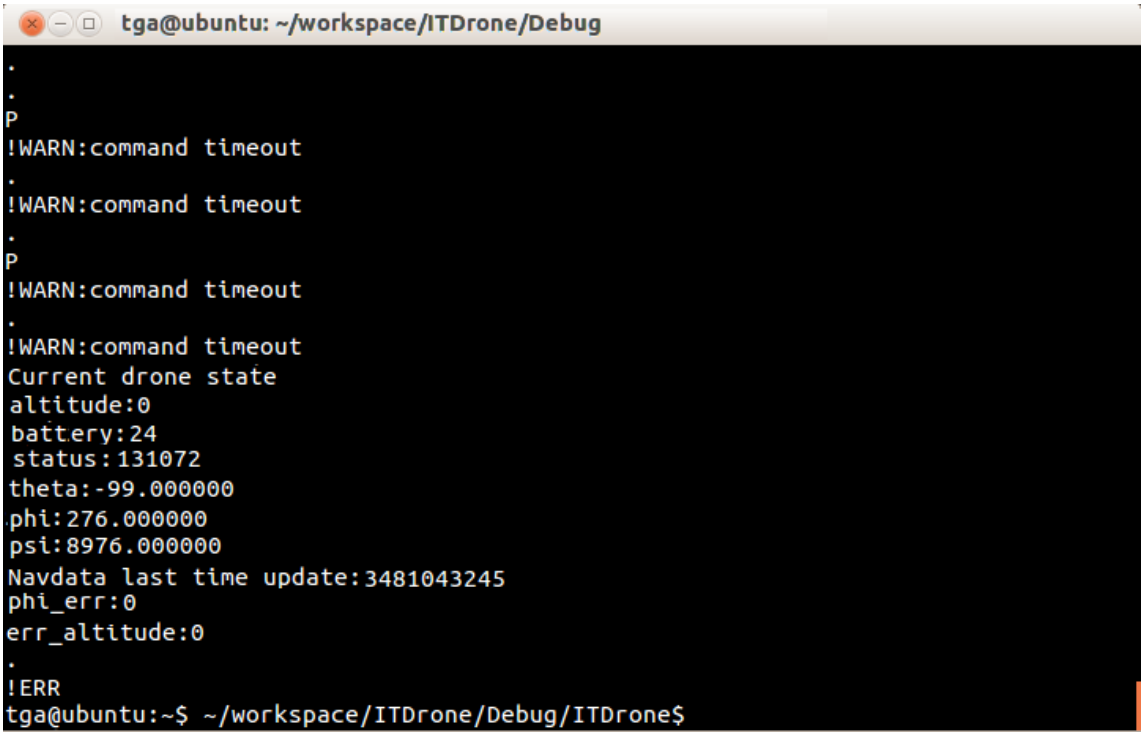

Figura 5.18 Janela de execução do controlador interno do *drone*

Os resultados foram satisfatórios, tendo o *drone* correspondido assertivamente aos comandos de controlo enviados através do terminal, como também foram recebidos os estados de navegação do *drone* quando requisitados através do comando *print*.

E, em terceiro lugar, foi acoplado, com o auxílio de uma placa branca, o recetor GPS ao *arduino*, estando este conectado via USB com o computador, como se visualiza na Figura A.3 presente em anexo. Através da montagem demonstrada foi possível realizar testes de fiabilidade e de precisão ao recetor GPS.

Como se pode ver na Figura 5.19, os resultados iniciais, com a configuração pré-definida do GPS, não foram os mais satisfatórios. A linha amarela representa o percurso realizado e os balões azuis, as coordenadas recebidas pelo recetor GPS.

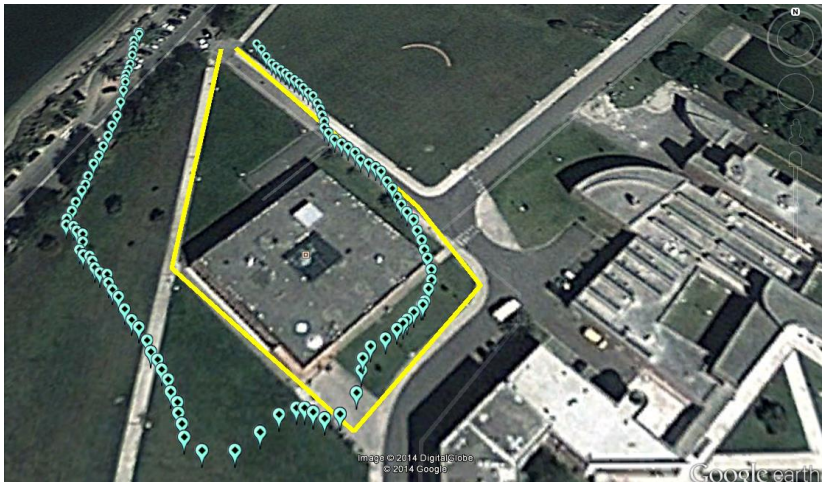

Figura 5.19 Leituras GPS a uma frequência de 1Hz

Com a modificação da frequência de receção de mensagens e da filtragem de mensagens apenas para os dois formatos pretendidos, verifica-se uma melhoria substancial na realização do mesmo percurso, como verificado na Figura 5.20.

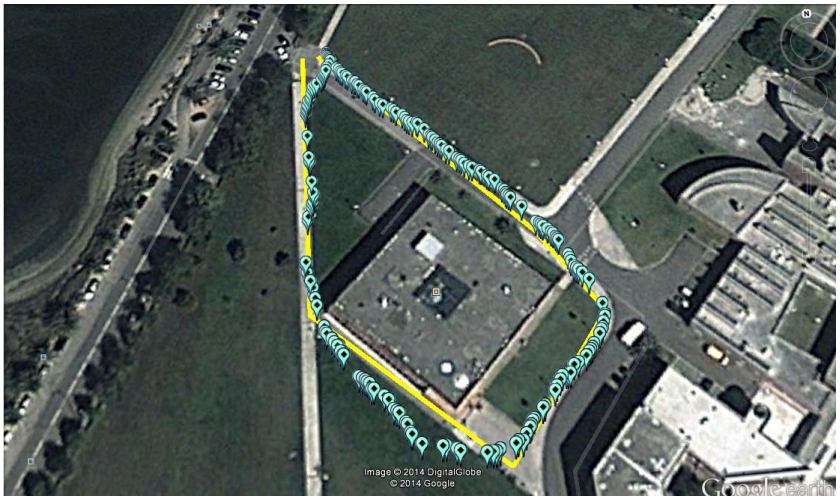

Figura 5.20 Leituras GPS a uma frequência de 5Hz

Deste modo, pode-se constatar que as coordenadas foram recebidas de uma forma muito mais precisa, aproximando-se, substancialmente, do percurso percorrido, ao invés do primeiro ensaio.

De seguida e após a realização dos testes de segurança e funcionamento dos blocos essenciais do sistema desenvolvido, são realizados vários voos de teste sendo que como resultado final obtiveram-se os resultados apresentados na Figura 5.21.

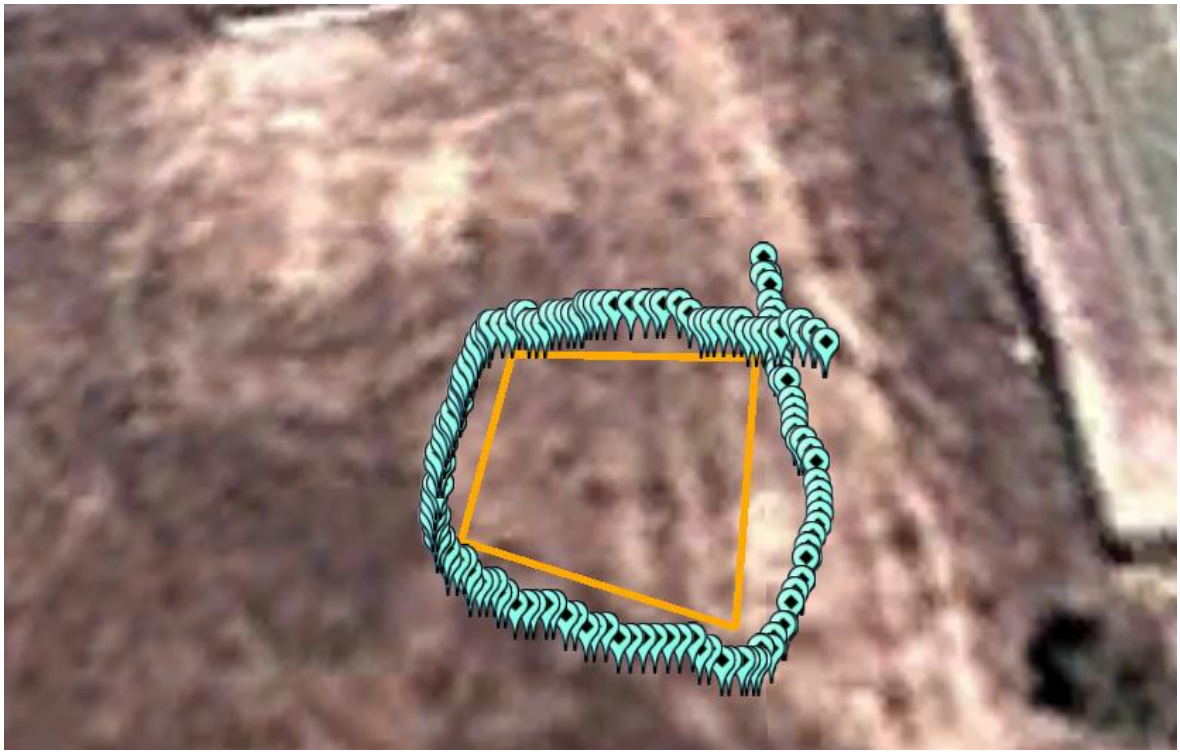

Figura 5.21 Resultados finais de voo

Como tal, verifica-se que a rota efetuada apesar de não ter seguido *in extremis* a rota pré-designada, devido à possível variação da precisão das coordenadas recebidas e das condições atmosféricas adversas ao voo, esta é bastante próxima do pretendido, sendo a sua variação de 1 a 3 metros da rota pretendida.

Conclui-se ainda, que existe uma variação da posição entre a posição de descolagem e a posição de aterragem, devido a inicialmente não serem captados os satélites suficientes para obter a precisão pretendida, sendo que quanto mais satélites captados, maior a precisão dos resultados.

Capítulo 6

Conclusões e Trabalho Futuro

Este capítulo final encontra-se dividido em duas secções: Conclusões e Trabalho Futuro.

As conclusões consistem num resumo geral do trabalho executado, dos resultados adquiridos e das possíveis limitações do sistema desenvolvido.

Na secção de Trabalho Futuro são apresentadas algumas evoluções viáveis, por forma a potenciar o sistema desenvolvido, e algumas eventuais soluções.

6.1 Conclusões

O principal objetivo deste trabalho foi desenvolver um sistema de navegação autónomo de um *drone* via GPS.

Numa primeira abordagem abordou-se a possibilidade de trabalhar sob o SDK disponibilizado pela empresa *Parrot*. No entanto, cedo se percebeu que não seria uma solução concretizável não só devido à escassa documentação sobre possíveis alterações no *firmware* interno desenvolvido pela empresa, mas também à forma pouco clara do funcionamento do SDK facultado.

Daí que, por existir a possibilidade de transferir sistemas desenvolvidos para controlar o *drone* via USB e correr o *software* desenvolvido através de um *cross-compile* interno, se tenha desenvolvido um sistema de navegação externo, com comunicação série com o *drone*.

Relativamente aos resultados obtidos, estes mostram-se ser satisfatórios, sendo possível realizar com sucesso a prova de conceito pretendida, tendo o resultado final uma pequena e espetável flutuação da rota realizada relativamente à rota pré-definida.

Os resultados preliminares foram também eles satisfatórios, tendo sido possível concluir e realizar uma comunicação com o melhor dos sucessos, tanto a nível de testes de fiabilidade do controlador interno processado no *drone* bem como da receção e processamento das coordenadas recebidas a partir do recetor de mensagens GPS.

Pretende-se que o trabalho desenvolvido represente uma contribuição para a evolução dos *drones* que potencialize a sua capacidade de operacionalização em vários cenários, auxiliando o utilizador na sua pilotagem.

De um ponto de vista inovador, prevê-se que a conjugação do sistema de navegação via GPS com outros sistemas de navegação autónomos contribua, num futuro próximo, para a evolução da aeronáutica em geral e dos meios de transporte.

6.2 Trabalho Futuro

O sistema desenvolvido, tal como qualquer sistema dependente de GPS, apresenta, no entanto, debilidades, em ambientes urbanos ou bastante arborizados e em ambientes *indoors*. A possibilidade de combinar sistemas de navegação via *GPS* com sistemas de *Simultaneous Localization and Mapping* (SLAM) auxiliados pelo IMU representa uma mais-valia em qualquer ambiente de navegação. Uma combinação entre navegação autónoma via GPS e as potencialidades da câmara frontal do *drone* resultaria num sistema de navegação com capacidade de operar em qualquer cenário. Este seria capaz de evitar obstáculos, tomar como referência outros objetos facilitando a navegação, corrigir possíveis desvios de navegação provocados por possíveis erros de leitura do GPS ou ainda auxiliar o sistema de navegação, caso este falhe.

É também imaginável a concretização de um aperfeiçoamento em termos de interface gráfica para o utilizador. O desenvolvimento de uma aplicação móvel, capaz de operar em *smartphones*, *tablets* ou computadores com a capacidade de introdução de novos *waypoints* quando o *drone* já se encontra em movimento, resultaria numa evolução considerável de navegação do *drone*. Outra possibilidade seria a de desenvolver uma interface gráfica, onde fosse possível, num único painel, a combinação entre a visão frontal do *drone* e o *tracking* em tempo real da rota efetuada pelo *drone*, proporcionando ao utilizador uma sensação de se encontrar num verdadeiro ambiente de um *cockpit*.

Anexos

A. Comandos Referentes à API do *AR.Drone 2.0*

A.1 Descrição de Comandos

AT*REF

O comando AT*REF controla o comportamento básico do *drone* (descolagem, aterragem e aterragem de emergência).

A sua sintaxe descrita na Tabela A.1 será:

AT*REF*=%d,%d<CR>*

- Argumento 1: número de sequência;
- Argumento 2: um valor inteiro no intervalo [0..2^{32}-1], representando o valor de 32 bits que controla o *drone*.

Tabela A.1 Especificação dos termos do comando AT*REF [25]

Bits	**31..10**	**9**	**8**	**7..0**
Significado	Não utilizar	*Takeoff/Land (“start bit”)*	*Emergency (“select bit”)*	Não utilizar

Os bits 18, 20, 22, 24 e 28 deverão estar a “1”. Os outros bits deverão estar a “0”.

Enviando um comando AT*REF com o bit 9 a “1” fará com que o *drone* efetue a descolagem (*Take off*).

Este comando deverá ser repetido até que o *drone* reporte através dos dados de navegação (*navdata*) que a descolagem ocorreu com sucesso.

Caso não receba qualquer outro comando após a descolagem, este entra em modo “*hovering*”, permanecendo numa posição o mais estaticamente possível a 1m do solo.

Aquando do envio do comando com o bit 9 a "0", o *drone* efetuará a aterragem. O comando deverá ser repetido até que a *navdata* devolvida pelo *drone* evidencie que a aterragem foi concluída com sucesso, não tendo sido reportada qualquer anomalia.

Após o primeiro comando "*start*", o *drone* permanece num estado "*taking off*", no entanto, admitirá outros comandos, o que significa que enquanto está a alcançar 1 metro de altitude e entrando consequentemente em modo "*hovering*", o utilizador poderá enviar comandos de movimento ou rotação.

Enviar um comando AT*REF com o bit 8 a "1", estando o *drone* em normal funcionamento, provocará um modo de emergência. Os motores entrarão em *stand-by*, independentemente do estado do *drone*, o que poderá originar impactos potencialmente violentos.

Quando o *drone* se encontra no modo de emergência deve enviar-se o comando com o bit 8 a "1" que fará com que o *drone* retome o estado normal. Não obstante, o *drone* somente deve retomar o voo, quando tenha garantias de que a circunstância que provocou a emergência tenha sido resolvida.

Enviar um comando AT*REF com o bit 8 a "0", fará com que o *drone* consiga receber possíveis comandos de emergência, prevenindo consecutivas ordens de emergência e de alternância do estado do *drone* entre emergência e estado normal. O exemplo de um comando de emergência enviado através de um pacote UDP:

***AT*REF**=1,290717696<CR>**AT*REF**=2,290717952<CR>**AT*REF**=3,290717696<CR>*

AT*PCMD

A função do comando AT*PCMD é a de enviar comandos de movimento para o *drone*.

A sua sintaxe descrita na Tabela A.2 Especificação dos termos do comando AT*PCMDTabela A.2 será:

***AT*PCMD**=%d,%d,%d,%d,%d,%d<CR>*

- Argumento 1: número de sequência
- Argumento 2: *flag* de permissão para o uso de comandos progressivos e/ou de modo combinado de *yaw*;
- Argumento 3: Inclinação do *drone* para a esquerda/direita (*roll*) – valor em vírgula flutuante entre -1 e 1.
- Argumento 4: Inclinação do *drone* para a frente/trás (*pitch*) – valor em vírgula flutuante entre -1 e 1.
- Argumento 5: Velocidade vertical (*gaz*) – valor em vírgula flutuante entre -1 e 1.
- Argumento 6: Velocidade angular (*yaw*) – valor em vírgula flutuante entre -1 e 1.

Tabela A.2 Especificação dos termos do comando AT*PCMD [25]

Bits	**31..3**	**2**	**1**	**0**
Significado	Não utilizar	*Absolute Control enable*	*Combined yaw enable*	*Progressive commands enable*

Este comando é reservado para controlar os movimentos do *drone*. A *flag* do bit 0 deverá permanecer sempre a "1" para que o *drone* considere os restantes argumentos. Configurando o bit a "0" fará com que o *drone* entre em modo "*hovering*".

A inclinação lateral (i.e. "*drone roll*" ou *Φ angle*) representa a percentagem da máxima inclinação do *drone*. Um valor negativo fará com que o *drone* incline e voe para a sua esquerda, um valor positivo fará com que este incline e voe para a sua direita.

A inclinação frontal (i.e., "*drone pitch*" ou *θ angle*) representa a percentagem da máxima inclinação configurada. Um valor negativo fará com que o *drone* baixe o seu "nariz", fazendo com que voe para a frente. Um valor positivo fará com que o *drone* eleva o seu "nariz", fazendo com que este voe para trás.

A velocidade de translação num plano horizontal depende de parâmetros externos não podendo ser determinada. Com os valores da inclinação frontal e vertical a zero, o *drone* estará fixo horizontalmente, no entanto, continuará a movimentar-se devido à inércia presente.

A velocidade vertical (i.e. "*gaz*") é a percentagem da máxima velocidade vertical definida. Um valor positivo fará com que o *drone* aumente a sua altitude relativamente ao solo. Um valor negativo fará com que este diminua a sua altitude.

A velocidade angular é a percentagem máxima de velocidade angular. Um valor positivo fará com que o *drone* gire para a direita. No sentido oposto, um valor negativo fará com que este gire para a sua esquerda.

AT*FTRIM

O comando AT*FTRIM opera como um comando de calibração dos controladores internos do *drone*, indicando-lhe que está pousado horizontalmente.

A sua sintaxe é:

***AT*FTRIM**=%d<CR>*

- Argumento 1: número de sequência

Este comando deverá ser enviado em cada inicialização do *drone*, assegurando que este está de facto situado num plano horizontal.

Não realizar este processo, poderá fazer com que o *drone* não adquira uma estabilidade automática quando se encontra a voar, tal como poderá não identificar qual a sua atual inclinação. Este comando, porém, não deverá ser enviado depois do *drone* levantar voo.

Ao receber este comando, o *drone* irá automaticamente ajustar as condições de navegabilidade dos controlos *pitch* e *roll*.

AT*CALIB

A função do comando AT*CALIB é de indicar ao *drone* para efetuar uma calibração do seu magnetómetro. Este comando deverá ser enviado quando o *drone* está em voo. Ao receber o comando, o *drone* irá calibrar o seu magnetómetro através de uma rotação em torno de si próprio.

A sua sintaxe é:

***AT*CALIB**=%d,%d<CR>*

- Argumento 1: O número de sequência;
- Argumento 2: Identificador do dispositivo a calibrar.

AT*CONFIG

O comando AT*CONFIG irá configurar uma opção que seja válida no *drone*. Existe um vasto conjunto de configurações possíveis a introduzir no *drone*.

A sua sintaxe é:

***AT*CONFIG**=%d,%s,%s<CR>*

- Argumento 1: O número de sequência;
- Argumento 2: O nome da opção a ser configurada, entre " ";
- Argumento 3: O valor da opção, também ele entre " ".

AT*CONFIG_IDS

O comando AT*CONFIG_IDS irá enviar os identificadores do próximo comando AT*CONFIG.

Num ambiente de multiconfiguração, o comando deverá ser enviado sempre antes de cada comando AT*CONFIG. A configuração apenas será aplicada caso o *ids* coincida com o atual *ids* do *drone*.

AT*COMWDG

O comando AT*COMWDG representa o *reset* da comunicação através de um *timer watchdog*.

A.2 Configurações de Multiutilizador

Nas configurações de multiutilizador existe a necessidade de serem gerados para a aplicação, sessão e utilizador, os respetivos identificadores. Para tal, estes serão gerados através de um código de deteção de erros, *Cyclic Redundancy Check* (CRC32 - 32 bits) [49]. O seu objetivo é de detetar alterações nos dados que possam ocorrer na transmissão. Através de uma divisão polinomial do seu conteúdo, os blocos de dados são analisados a partir de um *checksum* indexado no final da mensagem.

CUSTOM:application_id

Define o identificador da aplicação (gera as configurações da aplicação caso estas não existam, substituindo as anteriores).

Para remover uma aplicação é necessário imprimir o caráter "-" antes de introduzir o respetivo id (ex., *"-01234567"*). Para remover todas as anteriores, introduz-se "-*all*".

Exemplo de comando AT:

*AT*CONFIG=605,"custom:application_id","0a1b2c3d"*

CUSTOM:application_desc

Define a descrição da aplicação atual.

Exemplo de comando AT:

*AT*CONFIG=605,"custom:application_desc","ITDrone"*

CUSTOM:profile_id

Define o identificador do utilizador atual (gera o identificador caso este não exista, substituindo o anterior). O procedimento é análogo ao procedimento da configuração *application_id*.

Exemplo de comando AT:

*AT*CONFIG=605,"custom:profile_id"," 0a1b2c3d"*

CUSTOM:profile_desc

Estabelece a descrição do utilizador atual.

Exemplo de comando AT:

*AT*CONFIG=605,"custom:profile_desc","Alvane"*

CUSTOM:session_id

Define o identificador da sessão atual (gera o identificador caso este não exista, substituindo o anterior). O procedimento é uma vez mais análogo aos anteriores.

Exemplo de comando AT:

*AT*CONFIG=605,"custom:session_id"," 0a1b2c3d "*

CUSTOM:session_desc

Estabelece a descrição da sessão atual.

Exemplo de comando AT:

*AT*CONFIG=605,"custom:session_desc","Sessão 2014"*

A3. Configurações Gerais

Nas configurações gerais, são criadas todas as configurações do âmbito de navegabilidade.

GENERAL:navdata_demo

O *drone* tem a capacidade de transmitir uma versão reduzida dos dados de navegação para os seus utilizadores ou enviar toda a informação disponível, inclusive informação sobre *debugging* que é desprezável para o controlo.

Caso o parâmetro seja estabelecido a "*TRUE*", a versão reduzida irá ser enviada pelo *drone*. Caso o parâmetro seja estabelecido como "*FALSE*", toda a informação disponível será enviada.

Exemplo de comando AT:

*AT*CONFIG=605,"general:navdata_demo","TRUE"*

A4. Comandos de Controlo

Nos comandos de controlo, são definidas as configurações dos controladores internos do *drone*.

CONTROL:outdoor

A configuração *outdoor* irá definir o modo de operabilidade do drone, em que este se encontra (ou não) a voar num ambiente exterior. Através desta configuração o medidor da velocidade de vento irá ser ativado. Desta forma, sempre que se realizar um voo num ambiente *outdoor*, este controlo deverá ser ativado.

Exemplo de comando AT:

*AT*CONFIG=605,"control:outdoor","TRUE"*

CONTROL:control_yaw

Nesta opção de controlo é defino o valor máximo da velocidade de *yaw*, em radianos por segundo. Os valores recomendados deverão estar compreendidos entre 40 rad/s até 350 rad/s. Valores fora deste intervalo poderão originar instabilidade.

Exemplo de comando AT:

*AT*CONFIG=605,"control:control_yaw","3.0"*

CONTROL:control_vz_max

Nesta configuração é definida a velocidade máxima do *drone*, em milímetros por segundo. Os valores recomendados deverão estar dentro do intervalo de 200 mm/s a 2000 mm/s. Valores fora do intervalo poderão provocar instabilidade.

Exemplo de comando AT:

*AT*CONFIG=605,"control:control_vz_max","1000"*

CONTROL:euler_angle_max

Nesta configuração é definido o máximo ângulo de inclinação do *drone* em radianos dos ângulos dos controlos *pitch* e *roll*. O parâmetro está representado em vírgula flutuante entre 0 e 0.52 (i.e., 0º e 30º).

Exemplo de comando AT:

*AT*CONFIG=605,"control:euler_angle_max","0.25"*

CONTROL:altitude_max

Na opção de controlo em questão é definido o parâmetro da altitude máxima do *drone* em milímetros. Qualquer valor introduzido irá ser definido como altitude máxima, sendo que o sensor de pressão realiza medições a qualquer altitude. Valores típicos de altitude máxima serão 100 000 mm (100 metros do chão), no entanto deverão ser considerados por motivo de segurança valores abaixo dos 6m de altura.

Exemplo de comando AT:

*AT*CONFIG=605,"control:altitude_max","3000"*

CONTROL:control_level

Na opção de controlo em questão, é descrita a forma como o *drone* irá interpretar comandos progressivos provenientes do controlador:

- O bit 0 refere-se a um ativador global, que deverá permanecer ativo

- O bit 1 refere-se ao modo de *yaw* combinado, onde os comandos de *roll* são usados para gerar rotações combinadas entre *roll* e *yaw*. A sua função principal é a de um modo de controlo simplificado para jogos de corridas de *drones*.

Exemplo de comando AT:

*AT*CONFIG=605,"control:control_level","3"*

A.5 Controlo de Vídeo

Nos comandos de controlo de vídeo, são configuradas as formas de encapsulamento de vídeo, formas de transmissão, bem como os *codecs* selecionados, entre outros.

VIDEO:bitrate_control_mode

Através desta opção de controlo será possível ativar o controlo automático de *bitrate* da transmissão de vídeo. Ativando esta configuração, reduzir-se-á a largura de banda utilizada pela transmissão de vídeo em fracas condições de conexão Wi-Fi, reduzindo a latência dos comandos.

- *VBC_MODE_DISABLE* (0): irá indicar o valor de *bitrate* na configuração de vídeo, *video:max_bitrate;*
- *VBC_MODE_DYNAMIC* (1): irá indicar o valor de *bitrate* do vídeo que varia entre [250;*video:max_bitrate*] Kbps;
- *VBC_MANUAL* (2): irá indicar qual o *bitrate* da transmissão de vídeo a ser fixado pelo controlo *video:bitrate*.

Exemplo de comando AT:

*AT*CONFIG=605,"video_bitrate_ctrl_mode","1"*

VIDEO:video_on_usb

Através deste controlo, em modo ativo ("*TRUE*") e com uma *pen USB* com uma capacidade superior a 100MB conectada, a transmissão de vídeo será gravada na respetiva *pen*. Caso contrário a transmissão do vídeo será enviada para o controlador.

Exemplo de comando AT:

*AT*CONFIG=605,"video:video_on_usb","TRUE"*

VIDEO:video_channel

Nesta opção de configuração é definido o canal pelo qual o vídeo será enviado para o controlador.

As opções são:

- *0: ZAP_CHANNEL_HORI*;
- *1: ZAP_CHANNEL_VERT*.

Exemplo de comando AT:

*AT*CONFIG=605,"video:video_channel","2"*

VIDEO:video_fps

Nesta opção é definido o FPS da transmissão de vídeo. O valor máximo é 30 FPS.

Exemplo de comando AT:

*AT*CONFIG=605,"video:codec_fps","30"*

VIDEO:video_codec

Nesta opção de controlo é definido o *codec* utilizado na transmissão de vídeo do *drone*. Os possíveis formatos de *codec* são:

- *MP4_360P_CODEC* (valor do parâmetro: 128): Transmissão de vídeo com codificador (*software*) *MPEG 4.2*. Sem capacidade de gravação;
- *H264_360P_CODEC* (valor do parâmetro: 129): Transmissão de vídeo com codificador (*hardware*) *H264* configurado a 360p. Sem capacidade de gravação;
- *MP4_360p_H264_720P_CODEC* (valor do parâmetro: 130): Transmissão de vídeo com codificador (*software*) *MPEG 4.2*. *Stream* de gravação no modo de 720p com o codificador *H264*;
- *H264_720P_CODEC* (valor do parâmetro: 131): Transmissão de vídeo com codificador (*hardware*) *H264* configurado a 720p. Sem capacidade de gravação;
- *MP4_360p_H264_320P_CODEC* (valor do parâmetro: 136): Transmissão de vídeo com codificador (software) *MPEG 4.2*. *Stream* de gravação no modo de 320p com o codificador *H264*.

Exemplo de comando AT:

*AT*CONFIG=605,"video:video_codec","129"*

VIDEO:max_bitrate

Nesta opção de controlo é definido o máximo *bitrate* a que o dispositivo pode descodificar. É configurado como o mais alto limite para os valores de *bitrate* do *drone*.

Quando utilizado o modo de controlo de *bitrate* como "*VBC_MANUAL*", o parâmetro não é considerado. No modo de controlo "*VBC_MODO_DISABLE*", o *bitrate* é fixado com o valor máximo.

Exemplo de comando AT:

*AT*CONFIG="video:max_bitrate","1000"*

B. Pinos de Entrada do Recetor GPS

VCC

O Porto Vcc representa a fonte de alimentação do módulo, em que a tensão deverá manter-se entre 3.2V e 5.0V (tipicamente 3.3V). A tensão de *ripple* deverá ser mantida abaixo de $50mV_{pp}$.

ENABLE

O porto *Enable* deverá manter-se em aberto ou em V+ para ligar; V- para desligar o módulo:

- *Enable(V^+): 1.8 V ≤ V_{enable} ≤ VCC*
- *Disable(V^-): 0 V ≤ V_{enable} ≤ 0.25 V*

VBackUp

O porto *VbackUp* representa a entrada de alimentação do *chipset* do GPS para manter a correr o *Real Time Clock* (RTC) quando a alimentação principal é retirada. A tensão suportada deverá estar entre os 2.0 V e os 4.3 V. (tipicamente 3.0V). O porto deverá ser conectado para operações normais.

3D-Fix

O porto *3D-fix* é designado como *fix flag output*. Caso não seja utilizado, deverá ser mantido no ar.

Antes do 2D-fix, O porto deverá continuamente produzir uma comutação de sinal de saída, ou seja, uma saída a "1" num segundo e no seguinte, a "0", tal como é visível na Figura A.1.

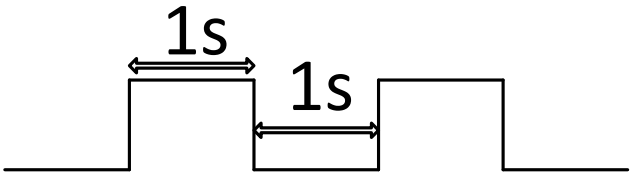

Figura A.1 Sinal precedente do *2D-fix*

Após o 2D ou 3D-fix, o porto deverá continuamente produzir um sinal a "0". Conectado ao porto estará um LED que estará a comutar até que seja adquirido o sinal de GPS.

TXD

O porto TXD representa o transmissor UART do módulo responsável pela transmissão de mensagens GPS para a aplicação.

RXD

O porto RXD representa o recetor UART do módulo utilizado para receber comandos e atualizações de *firmware*.

C. Ligações Físicas: Recetor GPS ⇔ *Arduino* ⇔ *Drone*

C.1 Esquema Elétrico do *Logic Level Converter*

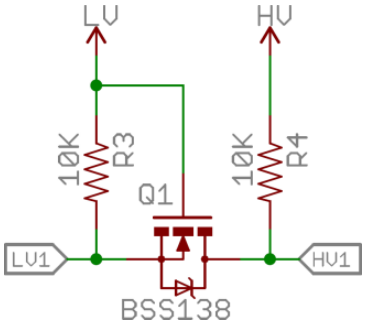

Figura A.2 Esquema Elétrico do *Logic Level Converter* [47]

C.2 Ligação *Arduino* ⇔ Recetor GPS

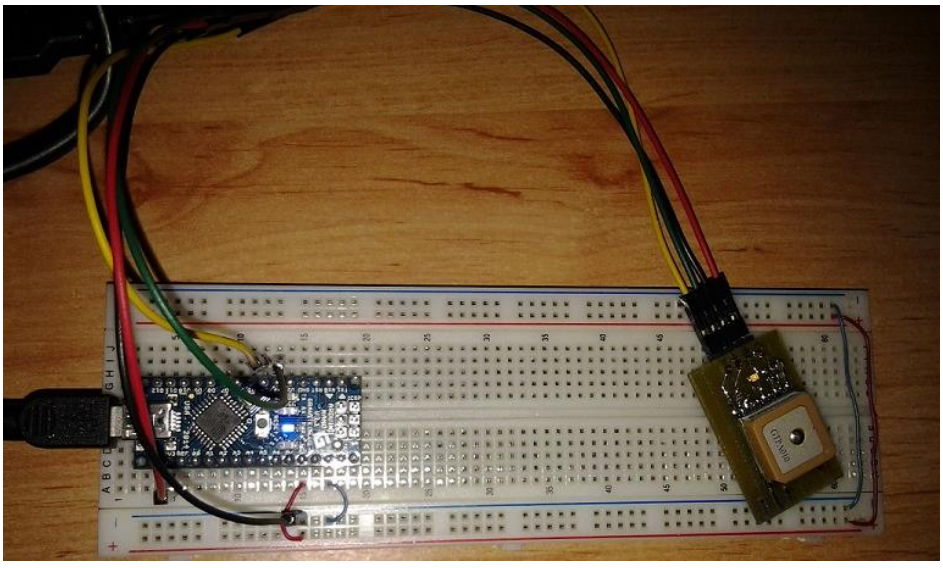

Figura A.3 Ligações físicas entre *arduino* e recetor GPS

Bibliografia

[1] "Send in the drones: Japan turns to robots to tackle Fukushima nuclear tragedy as officials admit plant could be leaking radiation for NINE months," Daily Mail, [Online]. Available: http://www.dailymail.co.uk/news/article-1377819/Japan-Nine-month-plan-seal-Fukushimas-nuclear-leaks.html. [Acedido em 20 01 2014].

[2] "Drone to help Oregon fight forest fires," Elko Daily Free Press, [Online]. Available: http://elkodaily.com/news/drone-to-help-oregon-fight-forest-fires/article_a39fe92a-fa87-11e3-997b-001a4bcf887a.html. [Acedido em 2014 30 02].

[3] "Microdrones-Applications: The sky is the limit," Microdrones, [Online]. Available: http://www.microdrones.com/en/applications/. [Acedido em 01 02 2014].

[4] "Quadcopter," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Quadcopter. [Acedido em 01 02 2014].

[5] "FAI Record ID #13093," Fédération Aéronautique Internationale, 30 04 2012. [Online]. Available: http://www.fai.org/fai-record-file/?recordId=13093. [Acedido em 01 10 2014].

[6] "FAI Record ID #13095," Fédération Aéronautique Internationale, 30 04 2012. [Online]. Available: http://www.fai.org/fai-record-file/?recordId=13095. [Acedido em 01 10 2014].

[7] "Étienne Oehmichen," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/%C3%89tienne_Oehmichen. [Acedido em 01 10 2014].

[8] "Parrot AR.Drone," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Parrot_AR.Drone. [Acedido em 01 02 2014].

[9] S. Khan, K. Ahmad, M. Murad e I. Khan, "International Journal of Computational Engineering Research," 07 2013. [Online]. Available: http://www.ijceronline.com/papers/Vol3_issue7/Part-3/G0373049054.pdf. [Acedido em 01 04 2014].

[10] B. Hofmann-Wellenhof, H. Lichtenegger e E. Wasle, "Introduction," em *GNSS - Global Navigation Satellite Systems*, Viena, Springer-Verlag, 2008, pp. 1-9, 309-310, 341-342, 365-366.

[11] U. A. Force, “Transit (Satellite),” Wikipedia, 04 08 2005. [Online]. Available: http://en.wikipedia.org/wiki/Transit_(satellite). [Acedido em 01 10 2014].

[12] “Uragan (GLONASS, 11F654),” GUNTER'S SPACE PAGE, 02 04 2014. [Online]. Available: http://space.skyrocket.de/doc_sdat/uragan.htm. [Acedido em 01 10 2014].

[13] ESA, “Galileo Satellite,” ESA, 08 12 2000. [Online]. Available: http://www.esa.int/spaceinimages/Images/2000/10/Galileo_satellite2. [Acedido em 01 10 2014].

[14] A. El-Rabbany, “Introduction to GPS,” em *Introduction To GPS: The Global Positioning System*, Norwood, MA, Artech House, 2002, pp. 1-11.

[15] U. A. Force, “Image Library,” GPS.GOV, 31 01 2014. [Online]. Available: http://www.gps.gov/multimedia/images/. [Acedido em 01 10 2014].

[16] U. C. G. N. Center, “GPS CONSTELLATION STATUS FOR 10/13/2014,” Navigation Center, 13 10 2014. [Online]. Available: http://www.navcen.uscg.gov/?Do=constellationStatus. [Acedido em 13 10 2014].

[17] P. Massatt e W. Brady, “Optimizing Performance Through Constellation Management,” *Crosslink,* vol. 3, n.º Orbit determination, pp. 17-21, 2002.

[18] U. A. Force, “New Civil Signals,” U.S. Government, 28 04 2014. [Online]. Available: http://www.gps.gov/systems/gps/modernization/civilsignals/#L2C. [Acedido em 01 10 2014].

[19] ESA, “GPS Future and Evolutions,” Navipedia, 2011. [Online]. Available: http://www.navipedia.net/index.php/GPS_Future_and_Evolutions. [Acedido em 01 10 2014].

[20] U. A. Force, “GPS Modernization,” GPS.Gov, 30 08 2014. [Online]. Available: http://www.gps.gov/systems/gps/modernization/. [Acedido em 01 10 2014].

[21] U. A. Force, “Control Segment,” U.S. Government, 17 09 2014. [Online]. Available: http://www.gps.gov/systems/gps/control/. [Acedido em 02 10 2014].

[22] “The Best GPS Device for Seniors,” SeniorTechDaily , 05 08 2013. [Online]. Available: http://seniortechdaily.com/the-best-gps-devices-for-seniors/. [Acedido em 05 10 2014].

[23] E. D. Kaplan e C. J. Hegarty, Understanding GPS: Principles and Applications, 2nd edition, Norwood, MA: Artech House, 2006.

[24] “What is EGNOS?,” ESA, 02 08 2013. [Online]. Available: http://www.esa.int/Our_Activities/Navigation/The_present_-_EGNOS/What_is_EGNOS. [Acedido em 05 10 2014].

[25] Parrot, ARDrone Developer Guide, França, 2012, p. 10.

[26] C. Anderson, “Parrot AR.Drones specs: ARM9, Linux, 6DoF IMU, Ultrasonics sensor, WiFi,” DIY DRONES, 06 01 2010. [Online]. Available: http://diydrones.com/profiles/blogs/parrot-ardrones-specs-arm9. [Acedido em 01 10 2014].

[27] R. Needleman, “Parrot AR.Drone quadcopter gets better specs and software,” Cnet, 08 01 2012. [Online]. Available: http://www.cnet.com/news/parrot-ar-drone-quadcopter-gets-better-specs-and-software/. [Acedido em 01 02 2014].

[28] “Parrot AR Drone 1.0 vs. Parrot AR Drone 2.0 comparison,” RC Quadrotors, 13 11 2012. [Online]. Available: http://rcquadrotors.com/parrot-ar-drone-1-0-vs-parrot-ar-drone-2-0-comparison/. [Acedido em 01 02 2014].

[29] “TECHNICAL SPECIFICATIONS,” Parrot AR.Drone 2.0, [Online]. Available: http://ardrone2.parrot.com/ardrone-2/specifications/. [Acedido em 01 02 2014].

[30] “Parrot AR.Drone 2.0,” [Online]. Available: http://ardrone2.parrot.com/photos/photo-album/. [Acedido em 01 08 2014].

[31] J. Messias, “Mobilis - Laboratório de Computação Móvel,” 03 09 2012. [Online]. Available: http://www.decom.ufop.br/imobilis/?p=1254. [Acedido em 01 02 2014].

[32] “New Mainboard for 2.0 AR.Drone,” ARDrone Flyers, 31 08 2013. [Online]. Available: http://www.ardrone-flyers.com/forum/viewtopic.php?f=8&t=6911. [Acedido em 01 02 2014].

[33] N. Dijkshoorn, “Hardware,” em *Simultaneous localization and mapping with the AR.Drone*, 2012, pp. 37-40.

[34] “Ultrasound,” Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Ultrasound. [Acedido em 01 08 2014].

[35] “ultrasound sensors,” AR.Drone Forum, 16 11 2012. [Online]. Available: http://forum.parrot.com/ardrone/en/viewtopic.php?id=6556. [Acedido em 01 08 2014].

[36] C. Anderson, “Parrot AR.Drone 2 announced, adds HD vid, baro,” DIY Drones, 08 01 2012. [Online]. Available: http://diydrones.com/profiles/blogs/parrot-ar-drone-2-announced-adds-hd-vid-baro. [Acedido em 01 02 2014].

[37] “UDP - User Datagram Protocol,” IPv6.com, [Online]. Available: http://ipv6.com/articles/general/User-Datagram-Protocol.htm. [Acedido em 01 02 2014].

[38] “O protocolo Telnet,” 10 2014. [Online]. Available: http://pt.kioskea.net/contents/286-o-protocolo-telnet. [Acedido em 20 10 2014].

[39] “IrDA and RS-232: A Match Made in Silicon,” maxim integrated, 01 10 2014. [Online]. Available: http://www.maximintegrated.com/en/app-notes/index.mvp/id/3024. [Acedido em 20 10 2014].

[40] P. W. Kahan, “Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic,” Berkeley CA, Californi, 1997.

[41] “NMEA 0183 - Standard For Interfacing Marine Electronic Devices, version 3.01,” 2002.

[42] Mediatek, “Mediatek - 3329 Datasheet Rev.A03,” 2010.

[43] “Arduino Nano,” Arduino, 2014. [Online]. Available: http://arduino.cc/en/Main/arduinoBoardNano. [Acedido em 10 10 2014].

[44] M. Robotics, “Arduino Nano 3,” [Online]. Available: https://www.robotics.org.za/index.php?route=product/product&product_id=88. [Acedido em 01 10 2014].

[45] M. Hart, “TinyGPS,” Arduiniana, 31 08 2013. [Online]. Available: http://arduiniana.org/libraries/tinygps/. [Acedido em 10 10 2014].

[46] C. Venesse, “Calculate distance, bearing and more between Latitude/Longitude points,” Movable Type Scripts, 2014. [Online]. Available: http://www.movable-type.co.uk/scripts/latlong.html. [Acedido em 02 10 2014].

[47] SparkFun, “SparkFun Logic Level Converter - Bi-Directional,” [Online]. Available: https://www.sparkfun.com/products/12009. [Acedido em 01 08 2014].

[48] F. Semiconductor, “Adafruit,” 10 2005. [Online]. Available: https://www.adafruit.com/datasheets/BSS138.pdf. [Acedido em 01 10 2014].

[49] M. Coder, “C Program to implement cyclic redundancy check CRC,” C Code Champs, 17 02 2013. [Online]. Available: http://www.ccodechamp.com/c-program-to-implement-cyclic-redundancy-check-crc/. [Acedido em 01 02 2014].

[50] “Control your own augmented reality aerial drone? There’s an app for that,” GizMag, 06 01 2010. [Online]. Available: http://www.gizmag.com/parrot-ardrone-iphone-controlled-remote-helicopter/13741/. [Acedido em 01 02 2014].

[51] M. Sullivan, “A brief history of GPS,” TechLive, 09 08 2012. [Online]. Available: http://www.techhive.com/article/2000276/a-brief-history-of-gps.html. [Acedido em 01 10 2014].

[52] D. K. Gunter, “BD 1A, 1B, 1C, 1D,” Gunter's Space Page, 25 03 2014. [Online]. Available: http://space.skyrocket.de/doc_sdat/bd-1.htm. [Acedido em 01 10 2014].